Table of Contents

Reference Architecture (RA) 1
Abstract 1
Book Contents 2
PDFs 2
Front Matter 4
a. Cover Page 5
OMG Discussion Paper Disclaimer 7
b. Change Log 8
c. Abstract 9
d. Copyright Notice 10
f. Preface 11
1 Introduction 13
1.1 Problem 15
1.2 Purpose 18
1.3 Content Organization 21
2 Architectural Views 23
2.1 Stakeholder Views 24
2.1.1 Platform View 27
Standards 27
Technical Standards 28
de facto Standards 28
2.1.2 Domain View 29
2.1.3 Ecosystem View 31
2.1.4 Ecosphere View 33
2.1.5 Exchange View 35
Standards 35
Technical Standards 35
de facto Standards 35
Tools 35
2.1.6 Enterprise View 37
2.1.7 Relevant Community Standards 40
Technical Standards 40
de facto Standards 40
2.2 Technical Views 42
2.2.1 Fundamental Views 43
2.2.1.1 Interfaces 44
Hardware 44
Software 44
Human 44
2.2.1.1.1 Platform Interface 46
Standards 46
Technical Standards 46
de facto Standards 46

Tools 47

2.2.1.1.2 Software Interfaces 48

Standards 48

Technical Standards 48

de facto Standards 48

Tools 48

2.2.1.1.3 Human Interfaces 50

Standards 50

Technical Guidance 50

de facto Guidance 50

Tools 51

2.2.1.2 Tools 52

2.2.1.2.1 Logging 53

Standards 53

Technical Standards 53

de facto Standards 53

Tools 53

2.2.1.2.2 Semantic Web 55

Standards 55

Technical Standards 55

de facto Standards 55

Tools 55

2.2.1.2.3 Open Source Communities 57

Standards 57

Technical Standards 57

de facto Standards 57

Tools 57

2.2.1.3 Case Management 58

Problems with Distributed Values (Domain Issues) 58

Problems with Distributed Software (Platform Issues) 58

Problems with Node 58

Summary 59

Standards 59

Technical Standards 59

de facto Standards 59

Tools 59

2.2.1.4 System of Systems (SoS) 61

Technical Standards 61

de facto Standards 62

2.2.1.5 Quality 63

Management 63

Modelling 63

Measurement 64

Requirements 64

Evaluation 64

2.2.1.6 Open Source Paradigm 66

Relevant Open Source Standards 66
Technical Standards 66
de facto Standards 66
2.2.1.7 Assurance 68
2.2.2 Node Network View 69
2.2.2.1 Network View 71
2.2.2.1.1 Secure Messaging 73
Standards 73
Technical Standards 73
de facto Standards 73
2.2.2.1.2 Transport 75
Standards 75
Technical Standards 75
de facto Standards 75
Tools 75
2.2.2.1.3 Security 77
Standards 77
Technical Standards 77
de facto Standards 77
Tools 77
2.2.2.1.4 Protocol 79
Standards 79
Technical Standards 79
de facto Standards 79
Tools 80
2.2.2.1.5 Distribution Software 81
Standards 81
Technical Standards 81
de facto Standards 81
Tools 81
2.2.2.2 Node View 83
2.2.2.2.1 Operating System (OS) 84
Standards 84
Technical Standards 84
de facto Standards 84
Tools 85
2.2.2.2.2 Operating Environment 86
Standards 86
Technical Standards 86
de facto Standards 87
Tools 88
2.2.2.2.3 DIDO Platform 89
Standards 89
Technical Standards 89
de facto Standards 89
Tools 89

2.2.2.2.4 Distributed Applications 91
Standards 91
Technical Standards 91
de facto Standards 92
Tools 93
2.2.2.3 Node Architecture 94
Common Core 96
2.2.2.3.1 Immutable Data Objects 97
2.2.2.3.1.1 Ledger 98
2.2.2.3.1.2 Transactions 99
2.2.2.3.1.3 Identities 100
2.2.2.3.1.4 Wallets 101
2.2.2.3.2 Ancillary Data 102
2.2.2.3.2.1 Journal 104
2.2.2.3.2.2 Transforms 105
2.2.2.3.2.3 Distributed Applications 106
2.2.2.3.2.4 Web Applications 108
2.2.2.3.2.5 Exchanges 109
2.2.2.3.3 Semantic Web 112
2.2.2.3.4 Software 113
2.2.2.4 Messaging View 115
Transactions 115
Transforms 115
Streams 115
2.2.3 Decentralized Finance (DeFi) Layers 117
2.3 Taxonomic Views 119
2.3.1 Network Topology Taxonomy 120
2.3.1.1 Centralized Network Topology 121
2.3.1.2 Decentralized Network Topology 123
2.3.1.3 Distributed Network Topology 125
2.3.1.4 Relevant Networking Standards 126
Technical Standards 126
de facto Standards 126
2.3.2 Network Access Control Taxonomy 127
2.3.2.1 Permissionless Networks 129
Benefits of Permissionless Networks 129
2.3.2.2 Permissioned Networks 131
Benefits of Permissioned Networks 131
2.3.2.3 Public Networks 133
Benefits of Public Networks 133
2.3.2.4 Private Networks 135
Benefits of Private Networks 135
2.3.2.5 Hybrid Networks 137
Benefits of Hybrid Networks 137

2.3.3 Node Taxonomy 139
2.3.3.1 Full Node 141
2.3.3.1.1 Pruned Node 142
Standards 142
Technical Standards 142
de facto Standards 142
Tools 142
2.3.3.1.2 Archival Node 144
2.3.3.1.2.1 Authority Node 145
2.3.3.1.2.2 Staking Node 146
2.3.3.1.2.3 Mining Node 147
2.3.3.1.2.4 Masternode 148
2.3.3.2 Lightweight Node (Wallet) 149
Standards 149
Technical Standards 149
de facto Standards 150
Tools 150
2.3.3.3 Lightning Node 152
2.3.3.4 Permanode 153
2.3.4 Data Taxonomy 154
Standards 154
Technical Standards 154
de facto Standards 154
Tools 154
2.3.4.1 Ledger Data 156
Standards 156
Technical Standards 156
de facto Standards 156
Tools 156
2.3.4.2 Ancillary Data 158
Examples of Ancillary Data 158
Supporting Data 158
Business Process Management 158
DIDO Implementation of Ancillary Data 159
Standards 159
Technical Standards 159
de facto Standards 160
Tools 160
2.3.4.3 External Data 161
Standards 161
Technical Standards 161
de facto Standards 161
Tools 161
3 Governance 162
Manage an Open Source Program 163

3.1 DIDO Communities 164
3.1.1 Stakeholder Communities 165
Steps for Establishing an Ecosphere 165
3.1.2 Software Communities 167
3.2 Legal Documents 168
3.2.1 Charter 170
Essential Elements of a Charter 170
Standards 170
Tools 171
References 171
3.2.2 Bylaws 172
Common Provisions in Bylaws 172
Standards 172
de facto Standards 172
Tools 173
References 173
3.2.3 Policies and Procedures (P&P) 174
Standards 174
Laws 174
Tools 174
References 175
3.3 Guides 177
4 Requirements 178
4.1 About Requirements 179
4.1.1 Governance Requirements Model 180
4.1.2 Cognitive Requirements Model 182
4.1.3 Governing Roles - Combined Requirements Model 184
4.1.4 Example of a Using the Combined Requirements Model 186
4.1.5 The Current State of DIDO Requirements 188
4.1.6 One Degree of Freedom Rule 190
4.1.7 Specifying Requirements 191
4.2 Functional Requirements 193
4.2.1 Platforms 194
4.2.1.1 Hardware Platform 196
About 196
4.2.1.2 Operating System Platform 197
4.2.1.3 Runtime Platforms 199
About 199
4.2.1.4 Network Platforms 201
4.2.1.5 Virtualized Nodes 202
About 202
Virtual Machines 203
Application Containers 204
4.2.2 Access Control 205

4.3 Non-Functional Requirements 211
About 211
Creating a Trade Study 211
4.3.1 Portability 213
About 213
4.3.1.1 Adaptability 214
About 214
DIDO Specifics 215
4.3.1.2 Installability 216
About 216
DIDO Specifics 218
4.3.1.3 Replaceability 219
About 219
DIDO Specifics 220
4.3.2 Reliability 222
About 222
DDS Specifics 223
4.3.2.1 Maturity 224
About 224
Products or Systems 224
Communities 225
DIDO Specifics 226
4.3.2.2 Availability 227
About 227
DIDO Specifics 228
4.3.2.3 Fault Tolerance 229
About 229
DIDO Specifics 230
4.3.2.4 Recoverability 231
About 231
DIDO Specifics 231
4.3.3 Maintainability 232
About 232
DIDO Specifics 233
4.3.3.1 Modularity 235
About 235
DIDO Specifics 236
4.3.3.2 Reusability 238
About 238
Types of Reuse 238
Types of Metrics and Models 241
DIDO Specifics 242
4.3.3.3 Analysability 243
About 243
DIDO Specifics 244
4.3.3.4 Modifiability 245

About 245
DIDO Specifics 248
4.3.3.5 Testability 249
About 249
DIDO Specifics 252
4.3.4 Securability 254
About 254
See DIDO Specifics 255
4.3.4.1 Confidentiality 257
About 257
DIDO Specifics 258
4.3.4.2 Data Integrity 259
About 259
DIDO Specifics 261
4.3.4.3 Non-Repudiation 262
About 262
DIDO Specifics 262
4.3.4.4 Authenticity 264
About 264
DIDO Specifics 264
4.3.4.5 Accountability 266
About 266
DIDO Specifics 266
4.3.5 Manageability 268
DIDO Specifics 268
4.3.5.1 Types of Manageability Functions 270
About 270
DIDO Specifics 271
4.3.5.2 Manageability Costs 273
About 273
DIDO Specifics 274
4.3.5.3 System Manageability Issues 276
About 276
Subsystem, Component and Module Lifecycle Issues 276
System Monitoring 277
System Logging 278
System Management 278
Vendor Lock-in Issues 280
DIDO Specifics 281
4.3.5.4 Software Manageability Issues 283
About 283
DIDO Specifics 284
4.3.6 Usability 285
About 285
Goals 285
Sub-Characteristics 285

Metrics 286

DIDO Specifics 286

4.3.6.1 Effectiveness Metrics 288

About 288

Using Task Completion Rates 288

Using Error Rates 289

DIDO Specifics 289

4.3.6.2 Efficiency Metrics 291

About 291

Time-Based Efficiency 291

Overall Relative Efficiency 292

DIDO Specifics 293

4.3.6.3 Satisfaction Metrics 294

About 294

DIDO Specifics 295

4.3.7 Performance 296

About 296

DIDO Specifics 296

4.3.7.1 Platform Performance 298

About 298

DIDO Specifics 299

4.3.7.2 Application Performance 300

About 300

DIDO Specifics 300

4.3.7.3 Network Performance 302

About 302

Speed 302

Wired Connections 303

Wireless Connections 303

Bandwidth 303

Network Quality of Service (QoS) 305

DIDO Specifics 305

4.3.8 Interoperability 307

About 307

Foundational or Technical Level 307

Syntactic Level 308

Domain or Structural Level 308

Semantic Level 308

DIDO Specifics 309

4.3.9 Elasticity 310

About 310

DIDO Specifics 310

4.3.10 Scalability 312

About 312

DIDO Specifics 312

4.4 Assessing Requirements 314

4.4.2 Non-functional Requirements Assessment 315
Assessing the Alternatives 315
The Vendor's Assessment 316
The Stakeholder's Focus Group 318
The Stakeholder's Assessment Team 319
The Stakeholder's Award Team 322
Appendices 324
Appendix A: Glossary of Terms Related to DIDO 325
!-* 326
0-9 327
16-Bit 328
32-Bit 329
64-Bit 330
A 331
Acceptance Testing 333
Access Control 334
Access Control Function 335
Access Control List (ACL) 336
Accessibility 337
Accountability 338
Adaptability 339
Address Resolution Protocol (ARP) 340
Address Resolution Protocol (ARP) Spoofing 341
After Action Review (AAR) 342
Aggregation Layer 343
Agile Model 344
American National Standards Institute (ANSI) 345
American Standard for Information Interchange (ASCII) 346
Analysability 347
Application 348
Application Container 349
Application Layer 350
Application Performance 351
Application Programming Interface (API) 352
Application Security 353
Application Specific Integrated Circuit (ASIC) 354
Appropriateness Recognizability 355
Architecture Adaptability 356
Argument 357
Assurance 358
Authentication 359
Authenticity 360
Authorization 361
Automation Pyramid 362
Availability 363
B 364
Backus–Naur Form (BNF) 365

Bandwidth ... 366
Big-Endian ... 367
Bill of Lading (BL or BoL) ... 368
Biometrics ... 369
Bitcoin Wallet ... 370
Bit Error ... 371
Black Box Testing ... 372
Blockchain ... 373
Blockchain Network ... 374
Block Producers ... 375
Block Validators ... 376
Bluetooth ... 377
Bootstrap ... 378
Bridge ... 379
Brownfield ... 380
Bylaws ... 381
Byzantine Fault Tolerance ... 382
Byzantine Generals Problem ... 383
C ... 384
Cable Subscriber Protection ... 386
California Consumer Privacy Act (CCPA) ... 387
Capability Maturity Model Integration (CMMI) ... 388
Category 5 (Cat-5) ... 389
Category 6 (Cat-6) ... 390
Category 7 (Cat-7) ... 391
Category 8 (Cat-8) ... 392
Central Processing Unit (CPU) ... 393
Charter ... 394
Children's Online Privacy Protection Act (COPPA) ... 395
Claim ... 396
Class ... 397
Client ... 398
Client-Server ... 399
Cloud Elasticity ... 401
Coins ... 402
Comma Separated Values (CSV) ... 403
Command Line Interface (CLI) ... 404
Command Shell ... 405
Commercial Off-The-Shelf (COTS) ... 406
Common Intermediate Language (CIL) ... 407
Common Language Runtime (CLR) ... 408
Common Object Request Broker Architecture (CORBA) ... 409
Communication Protocol ... 410
Communications Model ... 411
Community of Interest (CoI) ... 412
Compiler ... 413
Complex Instruction Set Computer (CISC) ... 414

Computer Architecture 415
Computing Platform 416
Conceptual Schema 417
Condition 418
Confidentiality 419
Confidentiality Agreement 420
Configuration Management (CM) 421
Conformance Specification 422
Consensus Algorithm 423
Container 424
Container Engine 425
Container Host 426
Container OS 427
Control Level 428
Copyleft 429
Copyright 430
Cryptocurrency 431
CyberSecurity Culture (CSC) 432
D 433
Daemon 435
Data-Centric 436
Data-Centric Publish-Subscribe (DCPS) 437
Data as a Service (DaaS) 438
Data at Rest 439
Database Driver 440
DataBase Management System (DBMS) 441
Data Definition Language (DDL) 442
Data Distribution Service (DDS) 443
Data in Motion 444
Data Integrity 445
Data in Use 446
Data Link Layer (DLL) 447
Data Logging 448
Data Manipulation Language (DML) 449
Data Model (DM) 450
Data Object (DO) 451
Data Protection 452
Data Protection Act 2018 453
Data Quality 454
Data Reader 455
Data Security 456
Datastore 457
Data Structure 458
Data Writer 459
DDS Domain 460
Decentralized Finance (DeFi) 461
de facto Standard 462

Delegated Byzantine Fault Tolerant (dBFT) 463
Delegated Proof of Stake (DPoS) 464
Department of Defense (DoD) 465
Dependent Event 466
DevOps 467
DIDO Domain Community 468
DIDO Ecosphere Community 469
DIDO Ecosystem Community 470
DIDO Platform 471
Digital Rights 472
Digital Rights Management (DRM) 473
Directed Acyclic Graph (DAG) 474
Disconnected, Intermittent and Limited (DIL) 475
Discovery 476
Disk Image 477
Distributed Application (ĐApp or DApp) 478
Distributed Denial-of-Service (DDoS) 479
Distributed Immutable Data Objects (DIDO) 480
Distributed Ledger Technology (DLT) 481
Distributed System 482
Docker 483
Domain Integrity 484
Domain Knowledge 485
Domain Name System (DNS) 486
Domain Participant 487
Download Speed 488
Downtime 489
Driver's Privacy Protection Act of 1994 (DPPA) 490
Duck Typing 491
Durability 492
Dynamic Host Configuration Protocol (DHCP) 493
Dynamic Link Library (.dll) 494
E 495
Economies of Scale 496
Elastic Compute Cloud (EC2) 497
Encryption 498
End-of-life (EoL) 499
End-to-End Solution (E2ES) 500
End-to-End Testing (E2E Testing) 501
Endianness 502
Endpoint 503
Enterprise Resource Planning (ERP) 504
Entity 505
Entity Integrity 506
Environment Variables 507
Ethereum Improvement Proposal (EIP) 508
Ethereum Request for Comment (ERC) 509

Ethernet 510
Evidence 511
Executable File 512
F 513
Failover 514
Fault Tolerance 515
Federal Deposit Insurance Corporation (FDIC) 516
Field Level 517
Fifty-One Percent (51% Attack) 518
FIGI Symbology 519
Figure of Merit (FoM) 520
File Transfer Protocol (FTP) 521
Financial Instrument Global Identifier (FIGI) 522
Firewall 523
Five Nines 524
Flowchart 525
Full-Disk Encryption (FDE) 526
Full Node 527
Functional Language 528
Functional Programming 529
Functional Requirements 530
Fungible 531
G 532
Gateway 533
General-Purpose Graphics Processing Unit (GPGPU) 534
General Data Protection Regulation (GDPR) 535
Goal 536
Google Mobile Services (GMS) 537
Government Off-The-Shelf (GOTS) 538
Graphical User Interface (GUI) 539
Graphics Processing Unit (GPU) 540
Greenfield 541
H 542
Hard Fork 543
Hardware Firewall 544
Health Insurance Portability and Accountability Act (HIPAA) 545
History QoS 546
Horizontal Scaling 547
Hub 548
Human-machine interface (HMI) 549
Hybrid Network 550
Hype-Cycle 551
Hypertext Transfer Protocol (HTTP) 552
Hypertext transfer protocol (HTTP) Request 553
Hypertext transfer protocol (HTTP) Response 554
Hypertext Transport Protocol Secure (HTTPS) 555
Hypervisor 556

I 557
Identification 559
Identifier (ID) 560
Immutable 561
Independent Event 562
Industrial Internet of Things (IIoT) 563
Information Assurance (IA) 564
Information Security (IS/InfoSec) 565
Information Technology (IT) 566
Infrared Wireless Networking 567
Infrastructure-as-a-Service (IaaS) 568
Initial Coin Offering (ICO) 569
Installability 570
Instance 571
Institute of Electrical and Electronics Engineers (IEEE) 572
Integration Testing 573
Intellectual Property (IP) 574
Interface 575
Interface Testing 576
International Organization for Standardization (ISO) 577
Internet 578
Internet of Things (IOT) 579
Internet Protocol (IP) 580
Internet Protocol address (IP address) 581
Internet Service Provider (ISP) 582
Interoperability 583
Interoperability Testing 584
ISO 9001 585
ISO 15288 586
ISO 90003-2018 587
J 588
Java Database Connectivity(JDBC) 589
Javascript 590
JavaScript Object Notation (JSON) 591
Jitter 592
Just-In-Time (JIT) 593
K 594
Key 595
Know Your Customer (KYC) 596
L 597
Latency 598
Learnability 599
Ledger 600
License Distribution 601
License Linking 602
License Modification 603
License Patent Grant 604

License Private Use 605
Licensing Sublicensing 606
Licensing Trademark Grant 607
Lightning Network 608
Light Node 610
Listener 611
Little-Endian 612
Liveliness 613
Local Area Network (LAN) 614
Logical Integrity 615
M 616
Maintainability 618
Maintainability Measure 619
Manageability 620
Management Level 621
Manufacturing Execution System (MES) 622
Massively Parallel Processing (MPP) 623
Mean Active Maintenance Down Time (MAMDT) 624
Mean Logistics Delay Time (MLDT) 625
Mean Time Between Failure (MTBF) 626
Mean Time To Failure (MTTF) 627
Mean Time To Repair (MTTR) 628
Media Access Control (MAC) 629
Message-Oriented Middleware (MOM) 630
Message Queue(MQ) 631
Metadata 632
Microcontroller 633
Micropayment Channel 634
Middleware 635
Miner Node 636
Mission Assurance (MA) 637
Mission Critical System 638
Modem 639
Modifiability 640
Modified Off-The-Shelf (MOTS) 641
Modularity 642
Module 643
Monitoring Software 644
Motherboard 645
Multi-core Processor 646
Multi-Signature (multisig) 647
Must (Requirement) 648
N 649
N-Tier Architecture 650
NATO Off-The-Shelf (NOTS) 651
Near-Field-Communication (NFC) 652
Network Appliance 653

Network Attached Storage (NAS) 654
Network Cabling 655
Network Device 656
Network Interface Card (NIC) 657
Network Layer 658
Network Node 659
Network Performance 660
Network Security 661
Network Topology 662
Network Traffic Analyzer 663
Node 664
Node Network 665
Non-Disclosure Agreement (NDA) 666
Non-Functional Requirements 667
Non-Repudiation 668
Normalization 669
NoSQL 670
O 671
Object 672
Object-Oriented Programming (OOP) 673
Objective 674
Object Management Group® (OMG) 675
One-Time PIN (OTP) 676
Ontology 677
Open Database Connectivity (ODBC) 678
Open Source Software (OSS) 679
Open Systems Interconnection (OSI) Model 680
Operability 681
Operating System (OS) 682
Operational transformation (OT) 683
Operator 684
Oracle 685
Owner 686
Ownership 687
P 688
Package Manager 690
Packet Loss 691
Packet Switched Network (PSN) 692
Parallel Processing 693
Parliamentary Authority 694
Partition 695
Payment Channel 696
Pedigree 697
Peer to Peer (P2P) 698
Performance 699
Performance Efficiency Measure 700
Performance or Functional Specifications 701

Permissioned Networks 702
Permissionless Networks 703
Permissive Open Source Software 704
Personal Identifiable Information (PII) 705
Physical Integrity 706
Physical Layer 707
Physical Security 708
Planning Level 709
Platform 710
Platform-as-a-Service (PaaS) 711
Platform Independent Model (PIM) 712
Platform Security 713
Platform Specific Model (PSM) 714
Plug In 715
Point-to-Point 716
Policies and Procedures (P&P) 717
Policy 718
Portability 719
Portable Operating System Interface (POSIX) 720
Port Number 721
Presentation Layer 722
Principle 723
Principles 724
Private Network 725
Privileges 726
Procedural Language 727
Procedure 728
Processor 729
Programming Language 730
Project Management Software 731
Proof of Authority (PoA) 732
Proof of Stake (PoS) 733
Proof of Work (PoW) 734
Protocol 735
Protocol Layer 736
Provenance 737
Public Key Infrastructure (PKI) 738
Public Network 739
Publish-Subscribe 740
Publisher 741
Q 742
Quality of Service (QoS) Policies 743
R 744
Radio Frequency Identification (RFID) 746
Read-Only Memory (ROM) 747
Reboot the World Problem 748
reCAPTCHA 749

Recoverability 750
Reduced Instruction Set Computer (RISC) 751
Reference Architecture (RA) 752
Referential Integrity 753
Registered Agent 754
Regression Testing 755
Relational DataBase Management System (RDBMS) 756
Reliability, Maintainability, and Availability (RAM) 757
Reliability Measure 758
Relocatable Object 759
Repairability 760
Repeater 761
Replaceability 762
Representational State Transfer (REST) 763
Request For Comment (RFC) 764
OMG 764
IETF 764
Request For Information (RFI) 765
Request For Proposal (RFP) 766
Requirement 767
RESTful API 768
Reusability 769
Rich Site Summary (RSS) 770
Risk 771
Roundoff Error 772
Router 773
S 775
Safety-Critical System (SCS) 777
Safety Assurance (SfA) 778
Salami Slicing 779
Sample 780
Sanity Testing 781
Sarbanes-Oxley Act (SOX) 782
Scalability 783
Scaling Out 784
Scaling Up 785
Script 786
Secure Shell (SSH) 787
Secure Sockets Layer (SSL) 788
Security Measure 789
Semantics 790
Semantic Web 791
Sensor 792
Sequence 793
Server 794
Servicability 795
Session Layer 796

Settlement_layer 797
Shall (Requirement) 798
Shared Library 799
Shared Object (.so) 800
Shielding 801
Should (Requirement) 802
Sidechain 803
Simple Payment Verification (SPV) 804
Single-Factor Authentication (SFA) 805
Six Sigma (6Sigma) 806
Smart Card 807
Smart Contract 808
Smoke Testing 809
Snapshot 810
Soft Fork 811
Software Adaptability 812
Software as a Service (SaaS) 813
Software Assurance (SwA) 814
Software Firewall 815
Software Library 816
Solution Stack 817
Source Code 818
Special Interest Group (SIG) 819
Special Rules 820
Sprint 821
Stakeholder 822
Standards Developing Organization (SDO) 823
Standards Organization 824
Standards Organization 825
Standing Rules 826
Static Library 827
Statute 828
Storage Device 829
Straight-through Processing (StP) 830
Structured Query Language (SQL) 831
Sub-Claim 832
Subject Matter Expert (SME) 833
Subscriber 834
Subscriber Identity Module (SIM) 835
Supervisory Control and Data Acquisition (SCADA) 836
Supervisory Level 837
Supply Chain 838
Switch 839
Symmetric Multiprocessing (SMP) 840
Syntax 841
System Assurance (SysA) 842
System Lifecycle 843
Systems and software Quality Requirements and Evaluation (SQuaRE) 844

T 845
Tangle 846
Taxonomy 847
Technical Standard 848
Testability 849
Throughput 850
Tokens 851
Topic 852
Total Cost of Ownership (TCO) 853
Transmission Control Protocol (TCP) 854
Transport Layer 855
Transport layer security (TLS) 856
Two-Factor Authentication (2FA) 857
U 858
Unicode Transformation Format (UTF) 859
Unified Modeling Language (UML) 860
Uninterruptible Power Supply (UPS) 861
Unique Identifier (UID) 862
Unit Testing 863
Universally Unique IDentifier (UUID) 864
Universal Serial Bus (USB) 865
UNIX 866
Upload Speed 867
Usability 868
Effectiveness 868
Efficiency 868
Satisfaction 868
Use Case 870
What is a use case? 870
Why are use cases important? 870
The purpose of use cases 870
User Datagram Protocol (UDP) 871
User Defined Integrity 872
User Error Protection 873
User Interface Aesthetics 874
User Scenario 875
V 876
Validation 877
Value Chain 878
Vendor Lock-In 879
Vertical Scaling 880
Video Privacy Protection Act (VPPA) 881
Virtual Disk Image (VDI) 882
Virtual Machine (VM) 883
Virtual Machine Images 884
W 885
WaitSet 886

Waterfall Model ... 887
Web Application (Web App) ... 888
Weight of Network ... 889
White Box Testing ... 890
Wide Area Network (WAN) ... 891
Will (Requirement) ... 892
Windows Registry ... 893
Wired Network ... 894
Wireless Fidelity (Wi-Fi) ... 895
Wireless Network ... 896
Wire Protocol ... 897
Wizard ... 898
X ... 899
eXtensible Markup Language (XML) ... 900
XML Information Set (XML Infoset) ... 901
XML Schema Definition (XSD) ... 902
Y ... 903
Z ... 904
ZigBee ... 905
Appendix B: Standards Organizations ... 906
Technical Standards Bodies ... 907
de facto Standards Bodies ... 908
Corporate Projects ... 908
Individual Projects ... 908
Appendix C: Hardware Architectures ... 910
C.1 Embedded Systems ... 912
C.1.1 Embedded Subsystem ... 913
About ... 913
DIDO Specifics ... 913
C.1.2 Standalone Embedded Systems ... 914
About ... 914
DIDO Specifics ... 914
C.1.3 Networked Embedded Systems ... 915
About ... 915
DIDO Specifics ... 915
C.2 Servers ... 916
About ... 916
DIDO Specifics ... 916
C.2.1 Software Servers ... 917
About ... 917
DIDO Specifics ... 917
C.2.2 Hardware Servers ... 919
About ... 919
DIDO Specifics ... 920
C.3 Desktops ... 921
About ... 921

DIDO Specifics 921
C.4 Handheld Computers 922
About 922
DIDO Specifics 922
C.5 Supercomputers 923
About 923
DIDO Specifics 923
C.6 Network Devices 925
About 925
DIDO Specifics 927
Appendix D: Operating Systems 929
Android 931
Apstra 933
Azure Real Time Operating System (or Azure RTOS) 935
Azure Sphere OS 936
balenaOS 937
Blackberry QNX 939
CentOS 940
Chromium OS 941
Cisco Digital Network Architecture (Cisco DNA) 942
Cisco Internetwork Operating System (IOS) 944
Cisco IOS XR 945
Cisco NX-OS 946
ClearOS 947
CloudReady 948
ExtremeXOS 949
FreeBSD 950
FreeRTOS 952
IBM i 954
iOS 955
Junos operating system (Junos OS) 956
LynxOS RTOS 957
Nokia X Software Platform 958
Open Network Linux 959
OpenServer 960
Oracle Linux (OL) 961
Oracle Solaris 963
Red Hat Enterprise Linux (RHEL) 964
SANtricity Software Operating System (OS) 965
SCO UnixWare 967
SUSE Linux Enterprise Server (SLES) 968
TrueNAS 970
Ubuntu Linux 971

Windows 973
Windows IoT 974
Windows NT 975
Windows Server 977
Windows XP 979
Appendix E: Tools 981
Community Tools 981
Network Traffic Analysis Tools 981
Tools for communications and collaboration 981
Tools for corporate-scale GitHub management 981
Tools: Archiving and Release Management 983
Tools: Bug and Issue Tracking 984
Tools: Code Reviews 985
Tools: Contributor License Agreements (CLA) 986
Tools: GitHub Management at Corporate Scale 987
Tools: Logging Tools 988
Tools: Network Traffic Analysis 989
Tools: Open Source Paradigm 992
Relevant Open Source Standards 992
Technical Standards 992
de facto Standards 992
Tools: Project Quality 994
Tools: Source Code Scanning and License Compliance 995
Tools: Tracking Project Health 997
Appendix F: DDS Quality Of Service 999
About 999
DDS Quality of Service Parameters 999
F.1 User Data 1001
F.2 Topic Data 1002
F.3 Group Data 1003
F.4 Durability 1004
F.5 Durability Service 1006
F.6 Presentation 1007
F.7 Deadline 1009
F.8 Latency Budget 1010
F.9 Ownership 1011
F.10 Ownership Strength 1012
F.11 Liveliness 1013
F.12 Time Based Filter 1015
F.13 Partition 1016
F.14 Reliability 1017
F.15 Transport Priority 1018
F.16 Lifespan 1019

F.17 Destination Order 1020
F.18 History 1021
F.19 Resource Limits 1022
F.20 Entity Factory 1023
F.21 Writer Data Lifecycle 1024
F.22 Reader Data Lifecycle 1025
Appendix G: Tests 1026
Appendix H: Acronyms 1029
Appendix I: Cognitive Model 1033
Bottom-Up Cognitive Model Example 1034
Top-Down Cognitive Model Example 1035
Appendix J: Governance Model 1037
Fundamental Governance Model 1037
Regulation 1038
Compliance 1038
Execution 1039

Reference Architecture (RA)

Return to Public Area **OR** View this document in the Wiki

DISTRIBUTED IMMUTABLE DATA OBJECT

(DIDO)

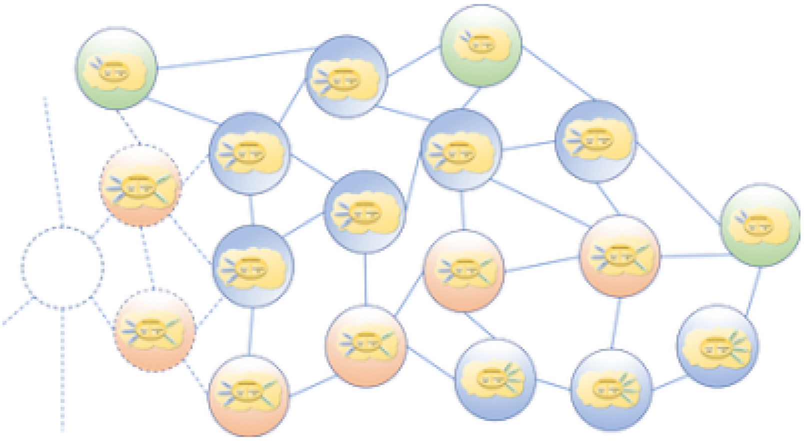
Reference Architecture (RA)

for

Cryptocurrencies, Blockchains, Distributed Ledgers and Tangles

Version 3.0

June 2021



R. W. "Nick" Stavros
Ian T. Stavros
Bryan Turek
Char Wales
Ian Murphy
Jackrabbit Consulting

Abstract

The excitement, expansion, and advancement in Distributed Computing resulting from Satoshi Nakamoto's paper "Bitcoin: A Peer-to-Peer Electronic Cash System"[1] highlights the need for a Reference Architecture (RA) focused on Distributed Immutable Data Objects (DIDOs), where it is understood that DIDOs include, but are not limited to cryptocurrencies, the original blockchain, and distributed ledger technologies. This paper, an OMG Discussion paper, presents an RA for DIDOs.

Book Contents

- Front Matter
- Table of Contents

- 1 Introduction
- 2 Architectural Views
- 3 Governance
- 4 Requirements

- **Appendices**
- Appendix A: Glossary of Terms Related to DIDO
- Appendix B: Standards Organizations
- Appendix C: Hardware Architectures
- Appendix D: Operating Systems
- Appendix E: Tools
- Appendix F: DDS Quality Of Service
- Appendix G: Tests
- Appendix H: Acronyms
- Appendix I: Cognitive Model
- Appendix J: Governance Model

PDFs

The following are provided for those readers who would like to download a printable albeit **static** copy of this RA, current as of the month/date shown.

- DIDO-RA 2.0 PDF (June 2020)
- DIDO Data Model (DIDO-DM) 1.0 PDF (June 2020)
- DIDO-RA 3.0 PDF (Slim Version - June 2021) This is a “slimmed down” version of the RA: it only contains the first pages of Appendices A through F. Any links to entries in these six Appendices will direct the reader back to the DIDO Wiki.
- DIDO-RA 3.0 PDF (Compact Version - June 2021) This version is much larger but still not the whole. It contains a truncated version of Appendix B, i.e., three pages only: “Appendix B”, “Technical Standards Bodies”, and “de facto Standards Bodies”. Any links to pages in Appendix B *other than these three pages* will direct the reader back to the DIDO Wiki.

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra

Last update: **2021/06/18 00:23**

Front Matter

return to Front of Book

- Cover Page
- OMG Disclaimer
- Change Log
- Abstract
- Copyright
- Contents
- Preface

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:0.front:start

Last update: **2021/06/08 18:25**

a. Cover Page

return to Reference Architecture (RA)

DISTRIBUTED IMMUTABLE DATA OBJECT

(DIDO)

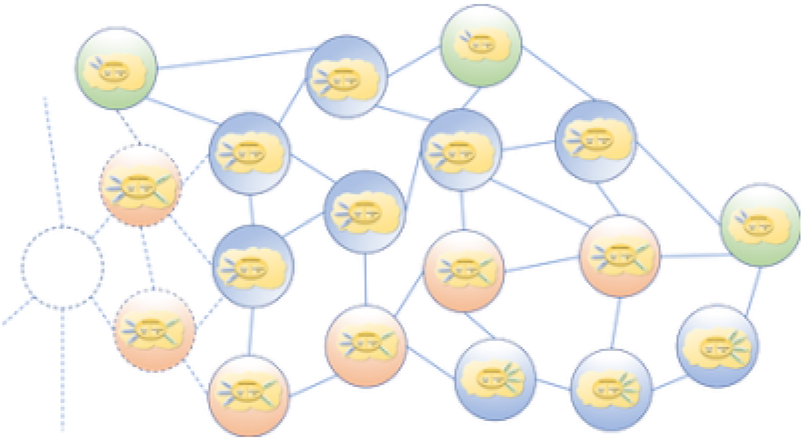
Reference Architecture (RA)

for

Cryptocurrencies, Blockchains, Distributed Ledgers and Tangles

Version 2.0

June 2020



R. W. “Nick” Stavros
Ian T. Stavros
Bryan Turek
Char Wales
Ian Murphy
Jackrabbit Consulting

This material is based upon work supported by the
U.S. Army Small Business Innovation Research Program Office,
Defense Advanced Research Projects Agency (DARPA) and the
Army Research Office under Contract Number: W911NF-17-P-0029.
The views and conclusions contained in this document or website
are those of the authors and
should NOT be interpreted as representing
the Official Policies, either expressly or implied,
of the
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY (DARPA)
or the
U.S. GOVERNMENT

OMG Discussion Paper Disclaimer

return to Front Matter

In accordance with the OMG's Policies & Procedures (P&P), to be issued as an OMG Discussion Paper, the DIDO RA 3.0 must include the following Disclaimer:

> *This paper presents a discussion of technology issues considered in a Subgroup of the Object Management Group. The contents of this paper are presented to foster wider discussion on this topic; the content of this paper is not an adopted standard of any kind. This paper does not represent the official position of the Object Management Group.*

b. Change Log

return to Front Matter

Changes made between version 2.0 and 1.0
The major change was moving the content from a Microsoft Word document to a Wiki. This transition allowed for content to be accessed on a section by section basis, easy cross-referencing of each section, and the ability to easily determine not only how many times a section is referenced, but also where. Specifically:

- Technical section
  o Reorganized into a series of technical views, citing relevant technical & de facto standards for each
  o As appropriate, tools to support technical views: added
- Taxonomy of data and nodes: added to Technical Views
- Governance for managing the DIDO Architecture: added
- Glossary of Terms (Appendix A): added
- A new section on technical and de facto standards (Appendix B): added
  o Data sheets for technical standards: added
  o Data sheets for de facto standards: added
- A new section for tools (Appendix C): added

Changes made between version 3.0 and 2.0

- Architectural Views & Governance (Sections 2 & 3): General clean-up and small additions, e.g., a new Technical View (Decentralized Finance (DeFi) Layers)
- Requirements (Section 4): added
- Glossary (Appendix A): expanded with additional terms
- Standards Organizations (Appendix B) - standards added and/or updated
- Hardware Architectures (Appendix C) - added
- Operating Systems (Appendix D) - added
- Tools (Appendix E) - no change
- DDS Quality of Service (Appendix F) - added, to be updated in 3.x
- Tests (Appendix G) - added
- Acronyms (Appendix H): expanded with additional entries
- Cognitive Model (Appendix I) - added
- Governance Model (Appendix J) - added

c. Abstract

return to Front Matter

> *The excitement, expansion, and advancement in Distributed Computing resulting from Satoshi Nakamoto's paper "Bitcoin: A Peer-to-Peer Electronic Cash System"*[2] *highlights the need for a Reference Architecture (RA) focused on Distributed Immutable Data Objects (DIDOs), where it is understood that DIDOs include, but are not limited to cryptocurrencies, the original blockchain, and distributed ledger technologies. This paper, an OMG Discussion paper, presents an RA for DIDOs.*

[2] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:0.front:2_abstract

Last update: **2021/06/08 19:11**

d. Copyright Notice

return to Front Matter

f. Preface

return to Front Matter

The Distributed Immutable Data Objects (DIDO) Reference Architecture (RA) is meant to be used as a resource to guide the design, use, or selection of Blockchain, Distributed Ledger Technology (DLT), or other Distributed Computing solutions such as InterPlanetary File System (IPFS) and Data Distribution Service (DDS).

The purpose of DIDO RA 1.0 was to create a better understanding of the Blockchain and DLT, which were exploding in the Information Technology (IT) world after the publication of Satoshi Nakamoto’s paper “Bitcoin: A Peer-to-Peer Electronic Cash System”[3] and the subsequent success of the Bitcoin. Since the publication of Nakamoto's paper, this excitement has grown way beyond the original Bitcoin. It has led to the promise/emergence of many other new cryptocurrencies, as well as the application of the well known and established concepts of distributed, peer-to-peer applications to supply chains, the Industrial Internet of Things (IIoT), natural resources, environmental sciences, and even the monetization of data.

In DIDO RA 2.0, the goal was to focus less on cryptocurrencies and more on generalizing peer-to-peer, distributed computing. As a parallel effort to the publication of DIDO RA 2.0, several products have been developed to work in parallel with and complement this paper:

- **DIDO Data Model (DIDO-DM)**: captures the conceptual data constructs described in the DIDO-RA including the Community of Interest (CoI) and testing.
- **DIDO Testing Environment (DIDO-TE)**: creates an environment that allows for virtualized testing of a Distributed Application (DApp) before it can be released into the “wild” using real hardware and networks.
- **DIDO Command Line Interface (CLI)**: defines a high level command language with which to send commands to each node covering the configuration, definition, and manipulation of data on distributed nodes.
- **DIDO Reference Implementation (DIDO-RI)**: provides a working interface to the DIDO-DM, DIDO-TE and DIDO-CLI.

The major enhancement of the DIDO RA 3.0 is the addition of a section on Requirements for DIDO Implementations from the “End User” perspective. Even though these requirements are for DIDO platform developers (e.g., Ethereum, Hyperledger, etc.) the intent is to enable a broad range of “End Users”, e.g., financial, retail, supply chains, etc., to employ one or more DIDO platforms. This new section defines a Governance Requirements Model, which is comprised of three processes: Regulations (i.e., requirements specification), Execution (i.e., lifecycle from design to maintenance), and Compliance (i.e., oversight of a product or service, as well as, the processes of Regulation and Execution). This model ensures DIDO implementations are well-governed and moves DIDO Governance from its current “Execution Centric” state (i.e, DIDO platform developers) to the overall DIDO Communities of Interest.

[3] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

1 Introduction

Return to Reference Architecture (RA)

The term Distributed Immutable Data Objects (DIDO) refers to the underlying technologies supporting distributed data and computation across a distributed network of peers using consensus algorithms to maintain integrity and consistency across the network. After the publication of Satoshi Nakamoto's paper *Bitcoin: A Peer-to-Peer Electronic Cash System*[2] and the exponential growth of other cryptocurrencies, there is a need to understand in general terms the underlying DIDO architectures and provide a framework to enable engineering due diligence of DIDOs. DIDOs are not limited to cryptocurrencies, the original blockchain, or distributed ledger technologies. DIDOs are also applicable to other non-cryptocurrency domains such as supply chain, registries for births, deaths etc., and lists of acceptable Identification and Authentication (IA) acceptable certificates, including those that have been revoked. The DIDO concepts captured within a Reference Architecture (RA) are intended to represent any architecture relying on distributed networks of peers that store data and allow parallel computation. The DIDO RA is not intended as a physical "must-have" requirements list, but more as a "what-can-be" conceptual catalog.

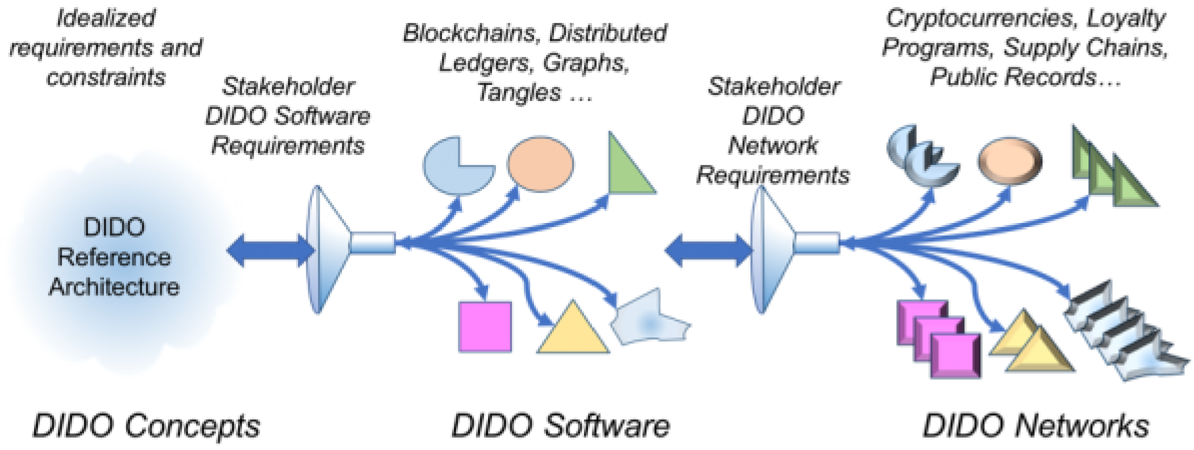

Idealized requirements and constraints
Stakeholder DIDO Software Requirements
Blockchains, Distributed Ledgers, Graphs, Tangles …
Stakeholder DIDO Network Requirements
Cryptocurrencies, Loyalty Programs, Supply Chains, Public Records…
DIDO Reference Architecture
DIDO Concepts
DIDO Software
DIDO Networks

Figure 1: The relationship of DIDO Reference Architecture to incarnations such as Bitcoins, distributed ledgers, and tangles.

As illustrated in the figure, the DIDO RA is an idealized general set of requirements and constraints. Each incarnation of the DIDO software (e.g., cryptocurrency or distributed ledger) uses a set of stakeholder requirements to filter the DIDO RA and tailor it to suit the unique needs and desires of its community of stakeholders. For example, the DIDO RA provides standards for logging, even though the logging requirements driven by DIDO software stakeholders may lead to a decision to opt not to include them. There are a plethora of DIDO software incarnations available, starting with the original Blockchain software that drives Bitcoin, and moving onto IBM's Distributed Ledger, Ethereum, and Iota.

The DIDO software incarnations are adopted or used by another community of stakeholders that wish to leverage the DIDO software into a DIDO Network of Nodes to address requirements and needs of a specific domain. For example, one DIDO network community wants to provide a cryptocurrency and another public records. The DIDO software selected by the DIDO network stakeholders might be different depending on its intended purpose. In Figure 1, the bi-directional arrows communicate the notion that the DIDO RA influences not only the DIDO software and networks, but also that evolution of DIDO software and networks feeds back into subsequent versions of the DIDO RA.

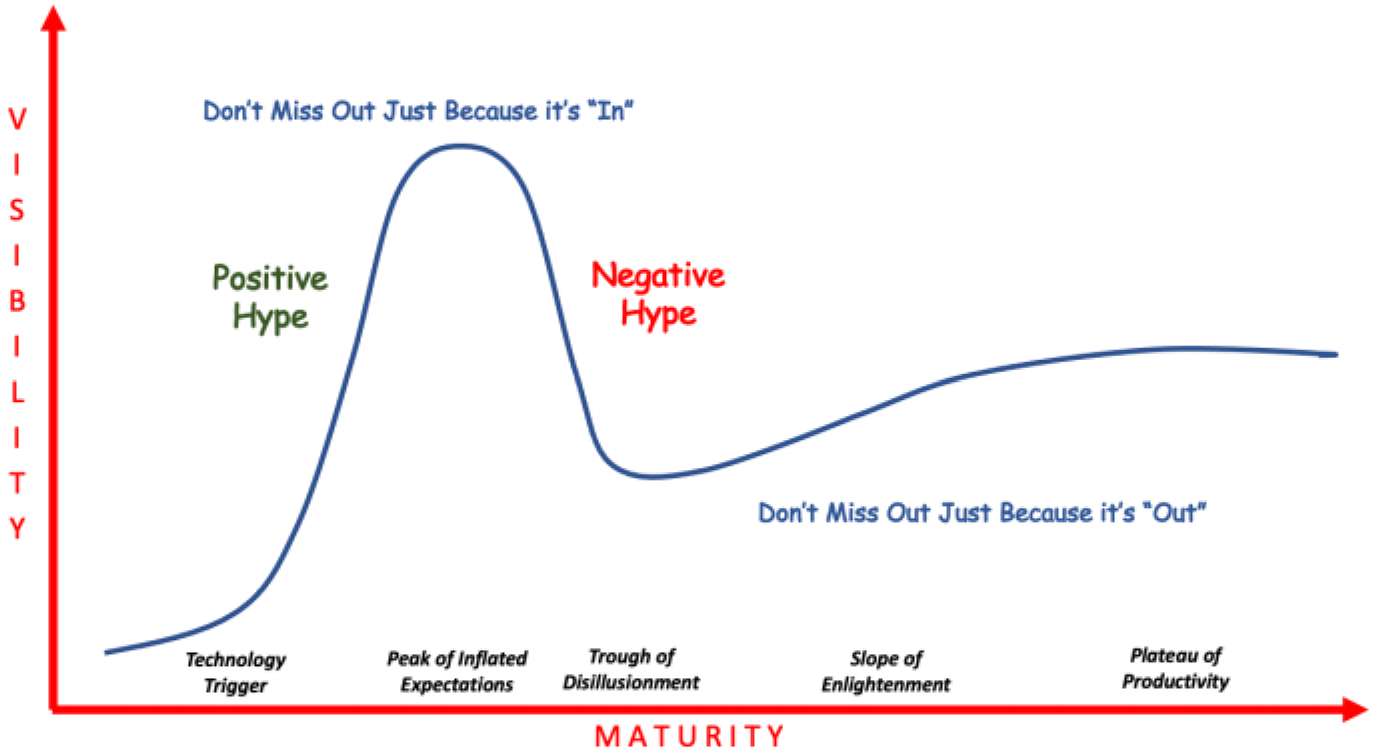
- 1.1 Problem
- 1.2 Purpose
- 1.3 Content Organization

[2] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

1.1 Problem

return to Introduction

As of June 12, 2018, there were 1,628 Cryptocurrencies available, with a total market cap around $300 Billion[5),6)]. DIDO implementations and interest in them are currently in the Positive Hype part of the technology Hype-Cycle as defined by Gartner.[7)]



VISIBILITY
Don't Miss Out Just Because it's "In"
Positive Hype
Negative Hype
Don't Miss Out Just Because it's "Out"
Technology Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
MATURITY

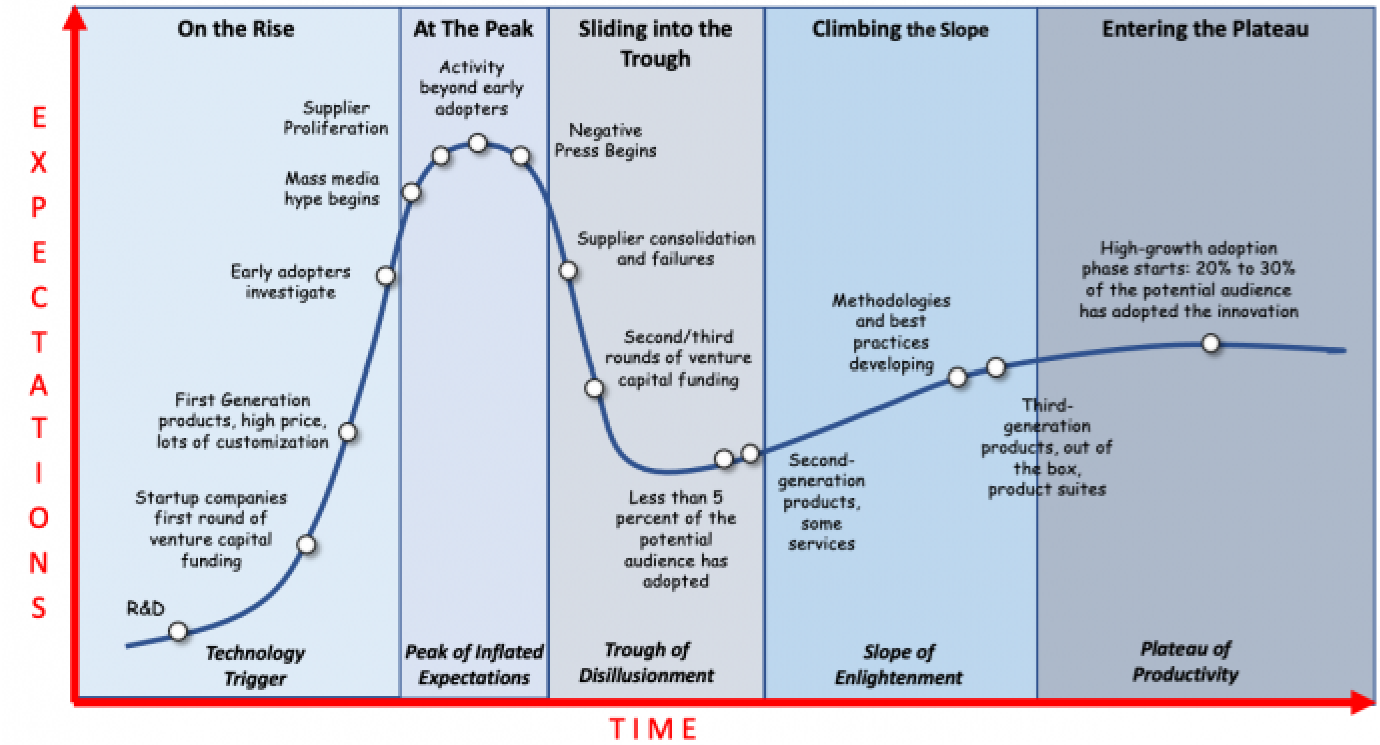


Figure 2: Gartner Group Hype Cycle

A major concern and part of the motivation behind the creation of a Reference Architecture (RA) is the lack of a comprehensive mechanism to evaluate all the risks associated with DIDO implementations (especially cryptocurrencies). Many of the risks to DIDOs are common to distributed computing and have already been identified and addressed using risk mitigators in existing processes, procedures, and standards. An example is vulnerabilities in source code. unfortunately, no statistics are available on how rigorously these risk mitigators are applied to DIDOs, if at all. Specifying a Reference Architecture and cross-referencing its components to a set of existing standards establishes the following:

- An actuarial framework for assessing risks associated with existing products
- Metrics for evaluating, comparing, and selecting existing products
- A roadmap for future standard implementation, enhancement, and development

[5] Coin Market Cap, “Cryptocurrency Market Capitalizations,” 3 December 2017. [Online]. Available: https://coinmarketcap.com/all/views/all/. [Accessed 3 December 2017].

[6] D. Palmer, “Market Cap Hits All-Time High,” 23 August 2017. [Online]. Available: https://www.coindesk.com/150-billion-total-cryptocurrency-market-cap-hits-new-time-high/. [Accessed 20 November 2017].

[7] Understanding Gartner's Hype Cycles, 30 May 2003, Alexander Linden, Jackie Fenn, https://www.bus.umich.edu/KresgePublic/Journals/Gartner/research/115200/115274/115274.html

1.2 Purpose

return to Introduction

The goals of a Reference Architecture (RA) as expressed in this paper are derived from the Office of the Assistant Secretary of Defense for Networks and Information Integration (OASD/NII)[8] , which are:

- Provide a common language for the various stakeholders
- Provide consistent implementation of technology to solve problems
- Support validation and comparison of implementations
- Encourage adherence to common standards, specifications, and patterns

Achieving these goals will enhance the likelihood that DIDOs and networks will be engineered correctly. DIDOs are intended to cover the entirety of distributed computing, including improvements in data storage and computing that are now used to support DIDO processes, but were not considered in Nakamoto's paper[9] and the consequent highly successful launch of Bitcoin cryptocurrency. Nakamoto's paper proposed a "ledger" for storing data and a collection of transactions, captured in a block. Blocks of transactions are verified and validated using a consensus algorithm. Bitcoin relies on Proof-of-Work (PoW) consensus. This solution, although usable, has proven to be expensive, making widespread ubiquitous adoption difficult[10]. Other implementations such as Ethereum were built upon the Bitcoin blockchain concept, which replaced the costly PoW with Proof of Stake (PoS). The DIDO RA is extensible to evolving and emerging blockchain technologies. For example, Boyen[11] notes that:

> *Our blockchain-free proposal shifts onto the transactions themselves the task of affirming prior transactions. Verification no longer results in a chain of transactions blocks, but in a lean graph comprised only of transactions…* [12]

The DIDO RA concerns distributed, Peer to Peer (P2P) computing including blockchains and cryptocurrencies, but also covers domains that have little or nothing to do with currencies. For example, DIDOs can be blockchains, distributed ledgers, or graphs. Blockchains may use any consensus algorithm such as PoW or PoS. Cryptocurrencies are used as a placeholder for any of the other domains that can be supported using DIDO: supply chains, government records, scientific data, medical records, escrows, swaps, etc.

The explosion in cryptocurrencies is well known and documented. An example is presented extremely well in the animation provided by Jeff Desjardins[13].

> **Note** "ICO funds raised" went from zero in 2014 to about $6.5 billion in November 2017.

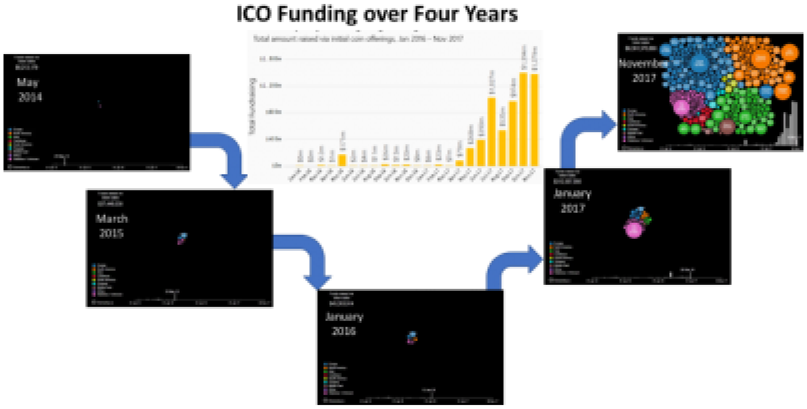


Figure 3: The explosive growth of Initial Coin Offerings (ICOs) over four years

To provide a common language for stakeholders, the DIDO RA describes the components of a distributed network of peers supporting distributed data and computation. It is comprised of a collection of peer nodes that operate within a virtual distributed network. Each node in the architecture selects the set of architectural components, and the relationships between the components, required by their stakeholders to solve the specific requirements (e.g., blockchains, cryptocurrencies, distributed ledgers, graph databases). The individual nodes synchronize via distributed software communicating over secure messaging infrastructure. All computations and operations could be executed redundantly on all the nodes within the DIDO network of peers. DIDO components are virtual representations of functionality found in DIDO products. In addition to identifying and defining the components and their inter-relationships, the RA associates each component with existing standards (refer to Section 2.1.7).

[8] G. Doyle and B. Wilezynski, “Reference Architecture Description,” U. S. DoD, Washington, DC, 2010.

[9] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[10] S. Lee, “Bitcoin's Energy Consumption Can Power An Entire Country – But EOS Is Trying To Fix That,” 2018 Aoril 2018. [Online]. Available: https://www.forbes.com/sites/shermanlee/2018/04/19/bitcoins-energy-consumption-can-power-an-entire-country-but-eos-is-trying-to-fix-that/#7a0603931bc8. [Accessed 11 June 2018].

[11] Gartner, “Gartner Hype Cycle,” [Online]. Available: https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp#. [Accessed April 2018].

[12] C. C. T. H. Xavier Boyen, “Blockchain-Free Cryptocurrencies, A Framework for Truly Decentralized Fast Transactions,” December 2017. [Online]. Available: https://eprint.iacr.org/2016/871.pdf. [Accessed 17 December 2017].

[13] J. Desjardins, “Business Insider - Visual Capitalist,” 13 December 2017. [Online]. Available: http://www.businessinsider.com/animation-shows-the-explosion-in-ico-funding-over-the-last-four-years-2017-12. [Accessed 17 December 2017].

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.1_intro:02_purpose

Last update: **2021/05/10 16:35**

1.3 Content Organization

return to Introduction

Following this Introduction, this discussion paper is organized as follows:

Section 2, Architectural Views: Presents and describes in detail the DIDO Architecture as a set of components collected into complementary Architectural Views: Stakeholder, Technical, and Taxonomic.

- **The Stakeholder View** is organized in accordance with the communities considered to be the "customers" of the DIDO RA, e.g., Platform, Domain, and Ecosystem.
- **The Technical View** is organized into two basic views: Fundamental (e.g., System of Systems, Assurance, Tools) and Node Network.
- **The Taxonomic View** is organized into four taxonomies: network topology, network access control, nodes, and data.

Standards (technical and *de facto*) relevant to or associated with the components supporting each view are listed.

Section 3, Governance: Proposes and describes the elements of the governance structure to manage the DIDO Architecture as an Open Source program, i.e., Charter (for the effort as a whole plus each of the DIDO COIs), Policies & Procedures (P&P), and Guide (plain English version of the P&P).

Section 4, Requirements: Lays the groundwork (models and methods) with which to specify, assess, manage, implement, and govern DIDO requirements to ensure well governed DIDO Implementations. Describes in detail the functional and non-functional requirements one must consider during the requirements specification process for DIDO platforms and then how to assess the resulting requirements specifications.

Appendices

- A: Glossary of Terms
- B: Standards Organizations: Complete listing and description of the Standards Organizations associated with the technical and *de facto* standards cited in the DIDO RA Architectural Views.
- C: Hardware Architectures : Describes Hardware Architectures to consider when designing distributed systems
- D: Operating Systems : Glossary of Operating Systems
- E: Tools : Tools to support the development, management, and operation of DIDOs
- F: DDS QoS: Quality of Service (QoS) parameters defined in the DDS specification
- G: Tests
- H: Acronyms
- I: Cognitive Model
- J: Governance Model

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.1_intro:3_organization

Last update: **2021/06/14 17:19**

2 Architectural Views

return to Reference Architecture (RA)

The DIDO Reference Architecture (RA) is presented as a series of components collected into different views. The components are then associated with a set of standards relevant to each component. There are two kinds of standards provided, each produced by different kinds of standards bodies:

- Technical Standards Bodies
- de facto Standards Bodies

There are three main categories of Architectural Views:

- 2.1 Stakeholder Views
- 2.2 Technical Views
- 2.3 Taxonomic Views

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:start

Last update: **2021/06/05 00:03**

2.1 Stakeholder Views

return to Architectural Views

Within this context, the following definition of Stakeholder is used:

> *A **stakeholder** is a person, group or organization that has interest or concern in a* [target] *organization. Stakeholders can affect or be affected by the* [target] *organization's actions, objectives and policies. http://www.businessdictionary.com/definition/stakeholder.html*
> **Note**: [target] added for clarification purposes

In a centralized or decentralized topography, this definition is adequate; however, in a distributed topography the understanding of what the *[target]* organization is becomes important. In a DIDO, this is by design and intent. There is no centralized authority or centralized cluster for the data, the processing of which is considered a major feature of the distributed architecture. It is a network of peers working together in parallel and simultaneously to solve problems. In other words, no single organization owns:

- all the computer resources or control them
- the definition of, or has control over, the processes
- the definition of data structures or data being distributed

In contrast to distributed systems, centralized systems (i.e., mainframes) are the authority for computation and data. In essence, the only reality is the processes and data that reside in the centralized system.

This is also in contrast to decentralized systems (i.e., traditional cloud servers), which rely on well-orchestrated and coordinated efforts of a few well connected and synchronized systems. Collective servers are the authority for the computation and data. In essence, the only reality is that which can be found on the decentralized servers; the infrastructure is expected to keep the software and data consistent and synchronized. However, this only needs to be concerned with servers that, although they are decentralized, still fall under a single authority thereby making the requirements, architecture, design, implementation, and maintenance relatively easy.

Both centralized and decentralized systems often have extensive data models and functionality, which adds to the complexity of managing them. This generally requires a single governing body (i.e., enterprise) to be ultimately responsible for the entire ecosystem and the lifecycle of the systems and the integration of components including hardware, operating systems, database management systems, web servers, application servers, software languages, networking, and other protocols. These “stacks” often result in stovepipe solutions.

In distributed systems, much of the ecosystem and governance of the components is handled by various Communities of Interest (CoIs): each has a responsibility for different aspects of the distributed system. The traditional role of a corporation or enterprise is to participate in these communities. The following graphic illustrates the various communities considered to be “customers” of the DIDO RA.

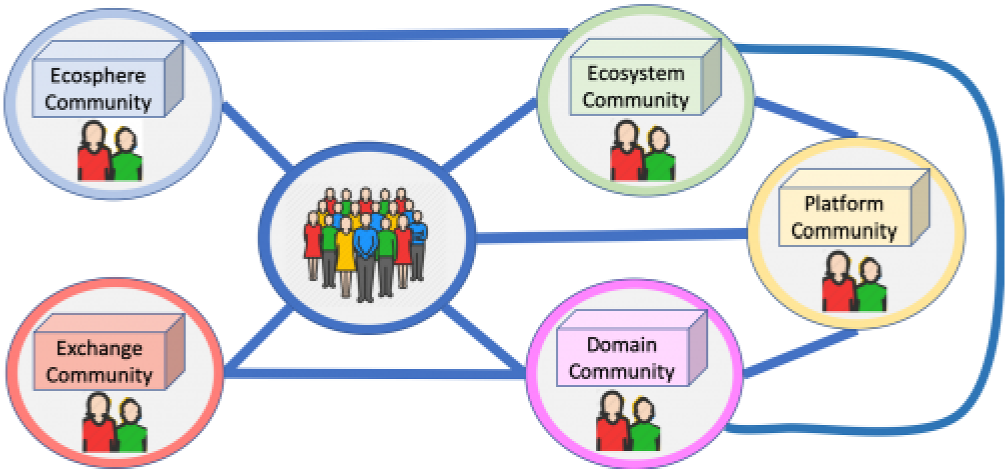


Figure 4: DIDO RA Stakeholder Communities

Platform

is responsible for the software used to distribute and control data within the Node Network, for example Bitcoin, Ethereum, Iota, DDS, and IPFS.

Domain

is responsible for the use of the data distributed on the Node Network, for example currency, rewards programs, or certificates.

Ecosystem

is responsible for a collection of domains associated with a particular ares of interest, such as green groceries, interest rate swaps, a particular tank, or class of automobiles.

Ecosphere

is responsible for a collection of domains and ecosystems associated with a common governance, which crosses over multiple areas of interest, such as military, government, automotive, or finance.

Exchange

is responsible for the exchange of data (tokens) from one domain or ecosystem with data in another domain or ecosystem, for example exchanging Bitcoins for U.S. dollars, or strawberries for jars of jam.

Enterprise

is responsible for being the systems integrator of all the domains, ecosystems, and ecospheres needed to fulfill the mission and goals of a corporation or organization, such as an auto company, a chain of retail stores, or a bank.

Each of these areas is explained in more detail in the following views, concluding with a list of standards applicable to these Stakeholder Views:

- 2.1.1 Platform View
- 2.1.2 Domain View
- 2.1.3 Ecosystem View
- 2.1.4 Ecosphere View
- 2.1.5 Exchange View
- 2.1.6 Enterprise View
- 2.1.7 Relevant Community Standards

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:1_stakeholder

Last update: **2021/05/10 17:22**

2.1.1 Platform View

return to Stakeholder Views

A Platform has a community of interest (CoI) that is responsible for the DIDO infrastructure that resides on each Node in the Node Network. Direct involvement within a platform covers the gamut from observers, to passive users of the software, and significant contributors to the community or de facto standards (see: 2.3.3 Node Taxonomy. As a general rule, domains, ecosystems, and ecospheres participate in platform communities rather than sponsor them. For the most part, platforms are already existing, self-contained communities with well established governance. Common CoI activities are bug tracking, collaboration and voting on resolutions of problems and future directions.

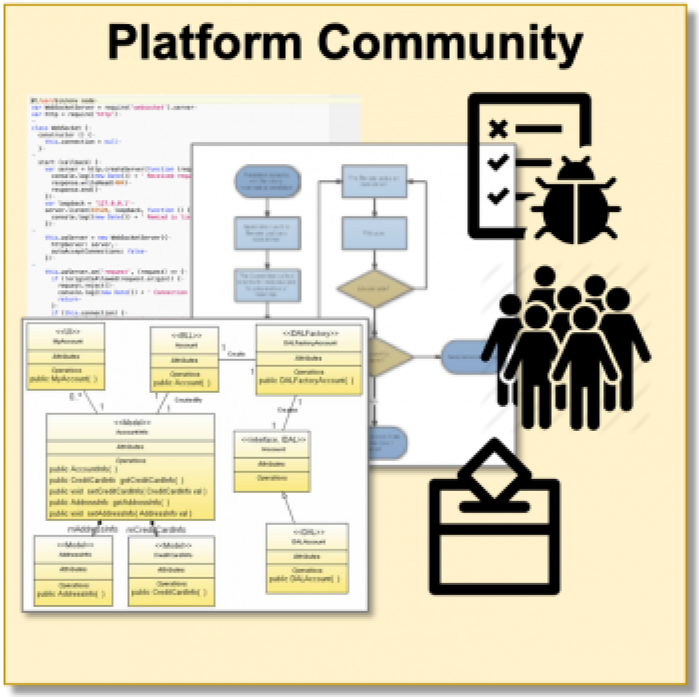


Figure 5: Platform Community

Some examples of platforms (see: 4.2.1 Platforms) are:

- Bitcoin
- DDS Foundation
- Ethereum
- Hyperledger
- Iota
- IPFS
- Multichain

Standards

Technical Standards

- RFC2026 - The Internet Standards Process
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)
- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)
- OMG: Interface Definition Language (IDL)

de facto Standards

- InterPlanetary File System (IPFS)
- Bitcoin
- Ethereum
- Linux Foundation: Hyperledger
- IOTA

2.1.2 Domain View

return to Stakeholder Views

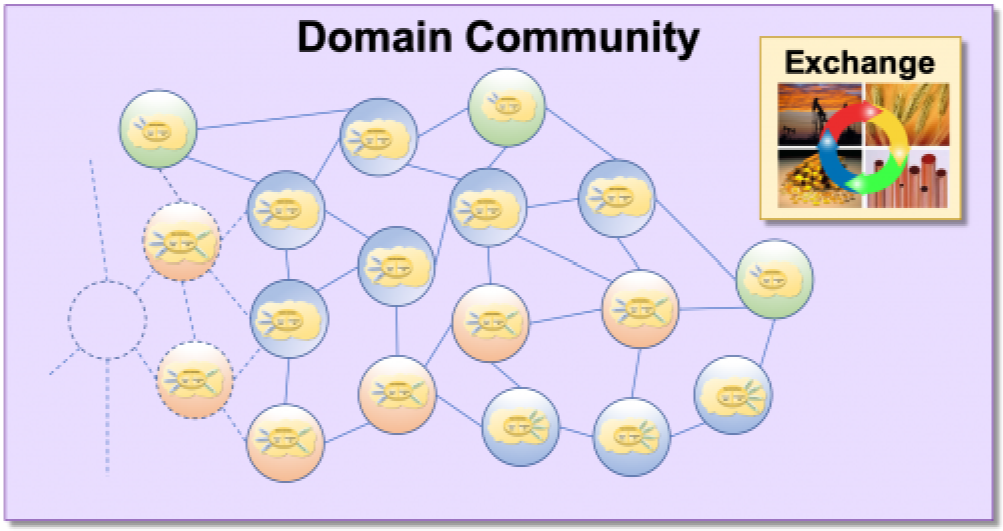


Figure 6: A Domain Community

The concepts of a **Domain View** are captured within the DIDO Domain Community.

A DIDO network consists of a system of nodes usually organized by a community of interest (CoI) connected together by a common, secure protocol – usually Transmission Control Protocol (TCP) and Internet Protocol (IP). Typically, each node executes its own copy of software that securely distributes data between the nodes. Data are generally contained within a transaction each node receives, interprets, and processes independently of the other nodes. Data can be as simple as base types such as integer, double, or string, or as complex as an entire document. All the data required to process the transaction may or may not be contained within the transaction or even the DIDO Network.

In most networks there is a one-to-one relationship between the system of nodes and fungible data managed by the network. For example, within the Bitcoin network, the *coin* is the Bitcoin. The system is solely dedicated to managing Bitcoin. In contrast, although Ether is the basic coin that drives the community and the system of nodes, Ethereum's software allows for multiple kinds of fungible data to be maintained within one network. For example, a single Ethereum network could manage Ether, customer loyalty reward points, and a registry of births and deaths.

Sometimes there is a need for multiple networks, each comprised of a different system of nodes. For example, there may be public networks or private (restricted) networks. Each network is centered on a particular kind of fungible data. Alternatively, the network might be aggregated into broader classifications such as those that have to do with currency (Ether, Bitcoin, etc.) and those that are based around customer loyalty reward programs (e.g., frequent flyer, guest, buyer programs). Still other networks might be based on public records (e.g., births, deaths, divorces), or commodities such as grains or metals. Governments or large corporations may decide to create private networks since the base of nodes they own and control is large enough for redundancy and security and can run on a private intranet. See: 2.3.2 Network Access Control Taxonomy

NOTE: Refer to 2.1.7 Relevant Community Standards for this view.

2.1.3 Ecosystem View

return to Stakeholder Views

The concepts of a **Ecosystem View** are captured within the DIDO Ecosystem Community.

The perspectives of Domain and Ecosystem Communities of Interest (CoIs) are different. Whereas a DIDO Domain Communityfocuses on a specific, real world solution or implementation to a specific concrete problem (i.e., a thing), an Ecosystem CoI focuses on the interactions and links between subordinate Ecosystems (i.e., sub-Ecosystems) and Domain Community. Governance of the overall enterprise Data Model (DM) and individual datastores for the enterprise should remain unchanged from the pre-DIDO or non-DIDO state; it is still hierarchical in nature with a definite chain of command and specific responsibility attributed to individuals within the organization. The primary focus of the DIDO Ecosystem is coordination of various other CoIs. For example, one Ecosystem CoI could coordinate other sub-Ecosystem and Domain Communities to ensure consistent perspective and context across for interoperability across all the CoIs.

A well run DIDO Community should have a charter that documents:

- the rules of engagement between DIDO CoI members
- the approval processes for changes to the charter and for releases, such as software and configuration
- trouble report procedures (capturing, distributing, resolving, disseminating afterwards)
- maintenance procedures (requesting, frequency, dissemination of maintenance reports)

- **Note** for the Ecosystem, the Charter and Bylaws are inherited from the encapsulating 2.1.3 Ecosystem View. Often, the Ecosphere is Charter, Bylaws and Policies and Procedures are subsets of the Ecosphere.

Ideally, a DIDO CoI should have a chairperson and a board of directors. Some DIDO CoIs may add a technical board or architectural board to oversee technical changes in the product and dedicate the board of directors to oversee governance of the DIDO CoI.

The following diagram represents the various kinds of DIDO CoIs and their relationship to each other.

- DIDO Ecosystem is comprised of 1 or more DIDO Platforms
- DIDO Ecosystem is comprised of 1 or more DIDO Domains
- DIDO Domain community is comprised of of 1 or more immutable data objects
- DIDO Exchange can bridge 1 or more immutable data objects within a DIDO community
- DIDO Exchange can bridge 1 or more DIDO communities
- DIDO Exchange can bridge 1 or more DIDO Ecosystems

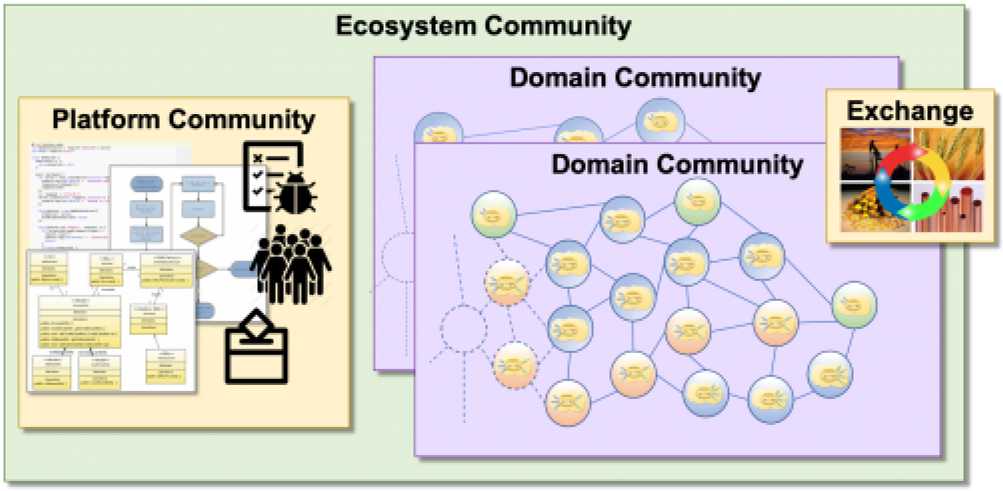


Figure 7: DIDO Ecosystem

NOTE: Refer to 2.1.7 Relevant Community Standards for this view.

2.1.4 Ecosphere View

return to Stakeholder Views

The concepts of a **Ecosphere View** are captured within the DIDO Ecosphere Community. The Ecosphere has legal requirements, see 3.2 Legal Documents.

The Ecosphere View comprises the set of all known Ecosystems within the Ecosphere and the interactions of the e=Ecosystems. Outside of a limited set of use cases, a DIDO cannot function independently, especially when a DIDO is ultimately meant to be part of an enterprise. The description of the DIDO Ecosystem Community works well when everything is defined as a DIDO using greenfield development. However, greenfield development for an enterprise probably will not happen, nor should it happen, given the large amount of legacy data information and processes held outside of the Ecosystem. Often, this type of development is referred to as brownfield development. This external data, referred to as *ancillary data*, **is** the immutable data object.

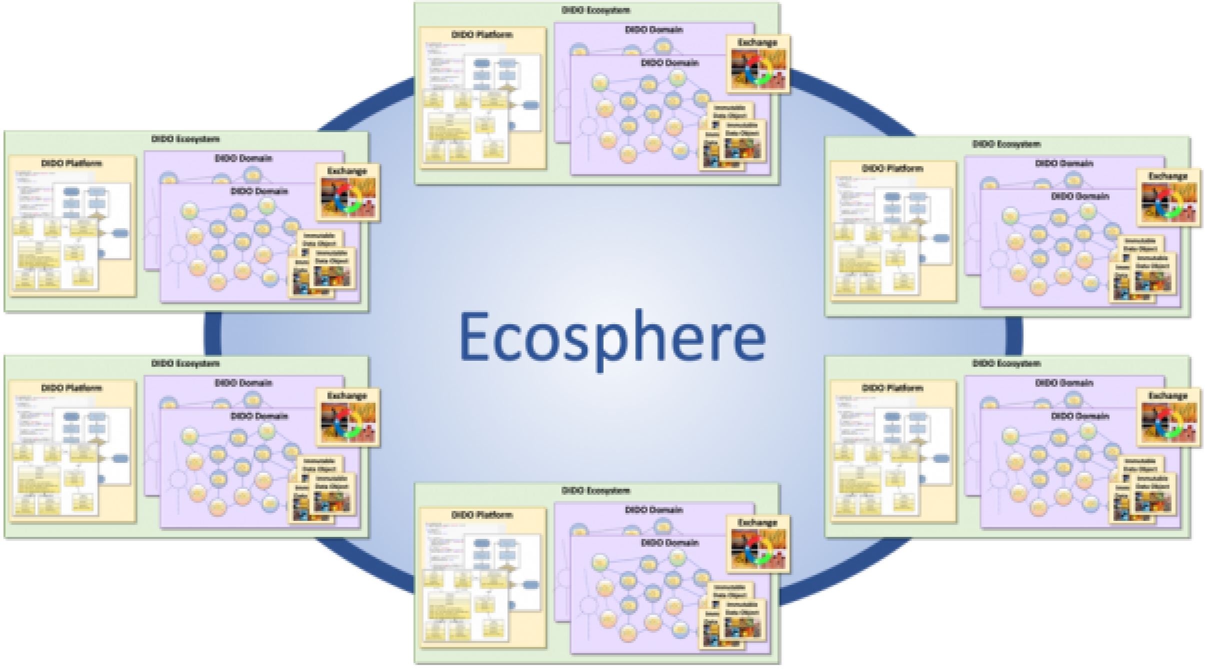


Figure 8: The high-level Ecosphere context

The Ecosphere concept is a way to encapsulate the Ecosystem and external ancillary data required to make the individual Domains within the Ecosystem functional. An individual DIDO Domain Community can access other data sources outside its Domain or, for that matter, even within its Ecosystem.

- **NOTE:** Refer to 2.1.7 Relevant Community Standards for this view.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:1_stakeholder:4_ecosphere

Last update: **2021/05/11 05:55**

2.1.5 Exchange View

return to Stakeholder Views

An exchange is responsible for the exchange of information between immutable data objects. An exchange can be wholly contained within a domain, act as a bridge between domains, or even span across ecosystems or ecospheres. Obviously, the complexity of an exchange increases as the range of what is exchanged moves from just internally within a domain and crosses ecosystem or ecosphere boundaries.

An exchange is a process that exchanges one kind of immutable data object for another. Bitcoin would not be worth much if there were no way to exchange it for existing currencies such as euro, dollar, etc. Exchanges may be completely contained within a DIDO community network or act as the bridge between networks within the community; or be used to transfer immutable data objects between communities (i.e., Bitcoins to Ethereum, Iotas, or commodities such as gold, silver and grains).

An example of an exchange that could operate completely within a community might be for customer rewards programs that include customer loyalty reward programs.

Standards

Technical Standards

- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- OMG: Ontology Definition Metamodel (ODM)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)

de facto Standards

- None at this time

Tools

- None at this time

2.1.6 Enterprise View

return to Stakeholder Views

Referring to the figure in Section 2.1 Stakeholder Views, one can observe that enterprises roughly map to an ecosphere; however, the enterprise does not necessarily control the ecosystems, platforms, or domains it now encompasses. From a DIDO perspective, all the ancillary data (i.e., external data) is accessed through the use of an *oracle*. There can be any number of oracles available to the node within a domain; these oracles can access data from centralized, decentralized, or other DIDOs. Some examples of oracles[14]:

- Another domain within the same ecosystem
- Documents
- Relational databases
- NoSQL databases
- Flat files
- Web services
- Application services
- Event logs
- Retail transactions
- Bank account records
- Stock market streams

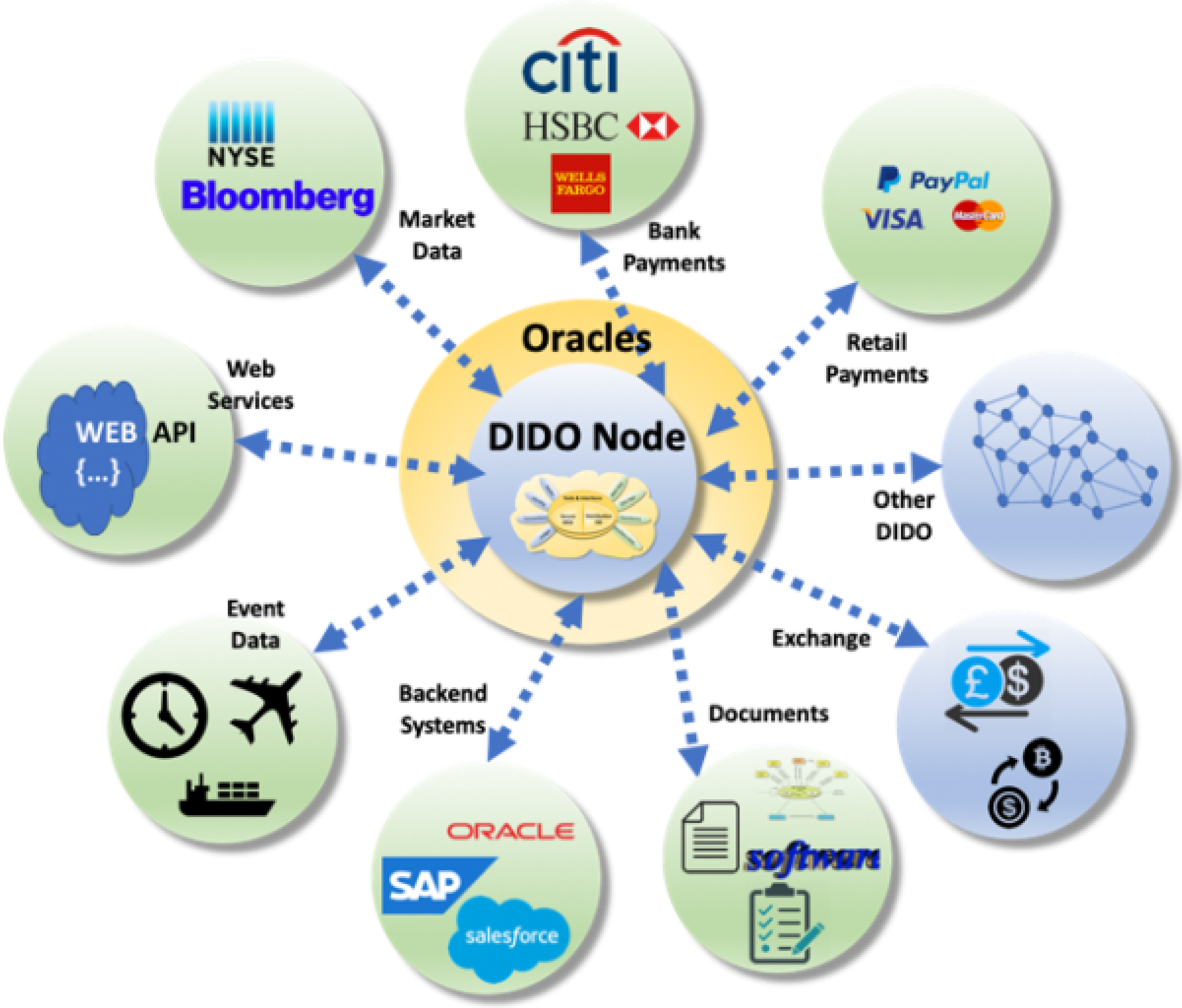


Figure 9: Oracles

One very important aspect of an enterprise with DIDO implementations is the heavy reliance on computer networks. These networks span all kinds of devices and operate in all kinds of environments, many of which rely on Disadvantaged Intermittent Links (DILs). Some examples are: nodes on mobile devices such as cell phones; submarines that are out of contact for long periods of time; nodes hit by natural and/or man-made disasters such as storms, earthquakes, or fire; devices that sleep to save power.

[14] This term should not be confused with the DBMS product from the Oracle Corporation; it is merely a term referring to something that is a good source of information.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:1_stakeholder:6_enterprise

Last update: **2021/05/10 17:22**

2.1.7 Relevant Community Standards

return to Stakeholder Views

The following standards are applicable to all stakeholder views.

Technical Standards

- RFC2026 - The Internet Standards Process
- ISO 9001:2015 Quality management
- ISO/IEC/IEEE 90003:2018 Software engineering – Guidelines for the application of ISO 9001:2015 to computer software
- ISO/IEC/IEEE 25000:2014 SQuaRE -- Guide to SQuaRE
- ISO/IEC 25001:2014 SQuaRE -- Planning and Management
- ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models
- ISO/IEC 25012:2008 SQuaRE -- Data Quality Model
- ISO/IEC 25020:2007 SQuaRE -- Measurement Reference Model and Guide
- ISO/IEC 25021:2012 SQuaRE -- Quality Measure Elements
- ISO/IEC 25022:2016 SQuaRE -- Measurement of Quality in Use
- ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality
- ISO/IEC 25024:2015 SQuaRE -- Measurement of Data Quality
- ISO/IEC 25030:2007 SQuaRE -- Quality Requirements
- ISO/IEC 25040:2011 SQuaRE -- Evaluation Process
- ISO/IEC 25041:2012 SQuaRE -- Evaluation Guide for Developers, Acquirers and Independent Evaluators
- ISO/IEC 25045:2010 SQuaRE -- Evaluation Module for Recoverability
- ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes
- OMG: Test Information Interchange Format (TestIF)

de facto Standards

- TODO: How to create an open source program
- TODO: Measuring your open source program's success
- TODO: Tools for managing open source programs
- TODO: Using open source code
- TODO: Participating in open source communities
- TODO: Recruiting open source developers
- TODO: Starting an open source project
- TODO: Improve your open source development impact
- TODO: Shutting down an open source project
- TODO: Building leadership in an open source community
- TODO: Setting an Open Source Strategy

2.2 Technical Views

return to Technical Views

Technical Views are intended to define at a high level the various components of larger DIDO systems. They outline generic parts, subparts, and interconnections of those parts and subparts for a DIDO system. For each part or subpart, a list of potential standards, best practices, and tools is provided. There are multiple views, subdivided into two main areas:

- 2.2.1 Fundamental Views
- 2.2.2 Node Network View

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views

Last update: **2021/05/10 17:22**

2.2.1 Fundamental Views

return to Technical Views

Fundamental Views are sets of standards that apply throughout all DIDO Ecospheres and are not tied to any specific DIDO component; they relate simply to good system and software engineering. These fundamental views are geared not just to software, but refer to other aspects of a DIDO lifecycle such as requirements, software quality, system and software assurance, and the specification of interfaces to the outside world.

There is no intended hierarchy in the following list of fundamental views.

- 2.2.1.1 Interfaces
- 2.2.1.2 Tools
- 2.2.1.3 Case Management
- 2.2.1.4 System of Systems (SoS)
- 2.2.1.5 Quality
- 2.2.1.6 Open Source Paradigm
- 2.2.1.7 Assurance

2.2.1.1 Interfaces

return to Fundamental Views

Interfaces exist between the various independent components of the DIDO and are therefore fundamental to the integration of DIDO components into a system. Bidirectional interfaces are defined between:

- hardware-to-hardware
- hardware-to-software
- hardware-to-human
- software-to-human

Hardware

Hardware interfaces are how the outside world connects to physical devices such as disk drives, monitors, keyboards, internal buses, batteries, internet ports, cameras, microphones, and sensors. These interfaces are described by the mechanical, electrical, and logical signals at the interface and the protocol for sequencing them.[15] Fortunately, most of these interfaces are integrated into computer nodes and do not need to be addressed here. For example, the data from the sensor can be considered an Immutable Data Object and thus sent through the node network, but the actual acquisition of that data is a detail for nodes to implement.

Refer to Section 2.2.1.1.1 Platform Interface for more detail on Hardware interfaces.

Software

Software interfaces isolate the underlying complexity of functionality provided by a software product or package through the use of an Application Programming Interface (API). The end result of this is to treat software products or packages as opaque blackboxes.

Refer to Section 2.2.1.1.2 Software Interfaces for more details on this subject.

Human

Human interfaces within DIDOs are bidirectional and primarily cover the software-to-human components. A software Application Programming Interface (API), although defining interactions between people and software, is classified as a software interface and not a human interface. Human interfaces cover the human interaction between the humans and an operational system; examples include mouse, keyboard, trackpad, and windows presented to users of the system.

Refer to Section 2.2.1.1.3 Human Interfaces for more details on this subject.

[15] Govindarajalu, B. (2008). “3.15 Peripheral Interfaces and Controllers - OG”. IBM PC And Clones: Hardware, Troubleshooting And Maintenance. Tata McGraw-Hill Publishing Co. Ltd. pp. 142–144. ISBN 9780070483118. Retrieved 15 June 2018.

2.2.1.1.1 Platform Interface

return to Interfaces

A **Platform** is the computing environment that software resides-in and executes-on. In the DIDO context, the software is the Distributed Application (ĐApp or DApp) that runs on a particular DIDO Platform (i.e., Bitcoin, Ethereum, IOTA, IPFS, DDS, etc). The DIDO Platform is comprised of the following abstractions:

1. the computer hardware (real or virtual)
2. the operating system (OS)
3. other layers of software used to support the application (DIDO Platform)

The first layer, the Operating System, abstracts access to the actual hardware such as memory, disks, networks etc. by providing Application Programming Interfaces (APIs) for applications to use. In other words, the application is tied to the OS rather than a particular hardware configuration. However, this still ties an application to an OS.

The second layer of abstraction occurs when the interface to the OS becomes standardized, as it is with the IEEE Portable Operating System Interface (POSIX) 1003.1-2016[16]. The POSIX standard supports applications portability at the source code level and includes provisions for a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs.[17]

The third layer (DIDO Platform) provides an API for a Distributed Application (DAPP) to reside-in and execute-on. Access to the other DIDO components is tightly controlled for security reasons. Unfortunately at this time, there is no standardized abstraction for DIDO Platforms.

Note: A DIDO Node within the DIDO Network provides a platform or subsetted platform so that DApps can be run on each node.

Standards

Technical Standards

- None at this time

de facto Standards

- Bitcoin: Developer's Guidance
- Bitcoin: Bitcoinj Developer's Documentation
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project

- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project

Tools

- None at this time

[16] IEEE Std 1003.1, 2016 Edition, IEEE Standard for Information Technology - Portable Operating System Interface (POSIX), includes IEEE Std 1003.1-2008, IEEE Std 1003.1-2008/Cor 1-2013, IEEE Std 1003.1-2008/Cor 2-2016.

[17] POSIX Product Standards, 1003.1-2016 Base, http://get.posixcertified.ieee.org/docs/base-2016.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:1_core:1_interface:1_platform

Last update: **2021/05/11 16:42**

2.2.1.1.2 Software Interfaces

return to Interfaces

Defining formal, well-formed, consistent, and uniform interfaces between nodes within the network and the interfaces between the components within a node is essential for providing stable products that can evolve over time. For example, the distributed nature of a DIDO means the system must be designed to account for inherent mismatches between versions of software running on each node within the network, especially since it takes time for updates to flow through the system (sometimes referred to as “reboot the world problem”).

Software interfaces are the most sensitive to component changes within the DIDO. The problem is similar to the logging problem, requiring cross platform, cross programming language requirements, and cross version support; not everything can be written in Java, Python, or even PHP.

Some other tools that might be useful in a distributed environment are debuggers, network analyzers, and discovery aids. Furthermore, given that the system supports all kinds of operating environments and multiple implementations, clear, unambiguous definitions of software interfaces are essential.

Formally defined and maintained Application Programming Interfaces (APIs) are necessary for all the DIDO parts to work together seamlessly.

Standards

Technical Standards

- OMG: Interface Definition Language (IDL)

de facto Standards

- None at this time

Tools

- None at this time

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:1_core:1_interface:2_software

Last update: **2021/05/11 16:42**

2.2.1.1.3 Human Interfaces

return to Interfaces

Until humans evolve to have one culture and speak with one language, there is no standard way to interface with humans. In many ways human interfaces are cultural and are dependent on the human experience. Human interfaces have evolved, and are evolving, so that the effectiveness of the interface is in many ways fickle, trendy, and highly subject to bias. However, this does not mean that there are no guiding principles for human interfaces.

Projects should have a Graphical User Interface (GUI) Style Guide and projects should remain consistent within those guidelines.

Note: Because of the cultural nature of user Interfaces, only *guidance* is listed, not standards.

Standards

Technical Guidance

- ADA Website Accessibility Under Title II of the Americans with Disability Act (ADA) - ADA Best Practices Tool Kit for State and Local Governments, Website Accessibility Under Title II of the ADA, Chapter 5, https://www.ada.gov/pcatoolkit/chap5toolkit.htm
- Accessibility of State and Local Government Websites to People with Disabilities, U.S. Department of Justice, Civil Rights Division, Disability Rights Section https://www.ada.gov/websites2.htm

de facto Guidance

Every project and program usually develops its own guidelines for user interfaces (UI). Often, these are modeled after some of the guidance offered by major corporations known for having “good” products.

- Android Developer, User Interface and Design, https://developer.android.com/guide/topics/ui
- Apple, Mac OS Developer, https://developer.apple.com/design/human-interface-guidelines/macos/overview/visual-index/
- Apple IOS Developer, https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/
- GNOME Human Interface Guidelines, https://developer.gnome.org/hig/stable/
- KDE Human Interface Guidelines, https://hig.kde.org/
- Microsoft Designing a User Interface, https://docs.microsoft.com/en-us/windows/win32/appuistart/designing-a-user-interface

Tools

There are many tools and frameworks available. The best advice is to remember that DIDO deployments are not as agile as stand-alone applications and require significant effort to keep the distributed applications in sync. Choose tools and frameworks keeping that in mind.

2.2.1.2 Tools

return to Fundamental Views

Tools are used to create, debug, maintain, or otherwise support other programs, applications, and communities. Within the DIDO architecture the application covers the DIDO platform, domain, exchange, node, and node network.

- 2.2.1.2.1 Logging
- 2.2.1.2.2 Semantic Web
- 2.2.1.2.3 Open Source Communities

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:1_core:7_tools

Last update: **2021/05/10 17:27**

2.2.1.2.1 Logging

return to Tools

One of the most critical requirements for distributed systems is the need to support common logging of events, information, and errors as well as support a multitude of programming environments (e.g., Java, C/C++, C#, JavaScript, Windows, UNIX, Linux).

> *Data logging is the process of collecting and storing data over a period of time in order to analyze specific trends or record the data-based events/actions of a system, network or IT environment. It enables the tracking of all interactions through which data, files or applications are stored, accessed or modified on a storage device or application.* [18)]

The protocol used for logging needs to be a redundant parallel system that reduces the dependency on the same infrastructure as the nodes and the node network to communicate. For example, if the node and node network use the same messaging software as the logging system, when a problem occurs, critical logging may also be down.

Standards

Technical Standards

- ISO 10003:2018 Quality management — Customer satisfaction — Guidelines for dispute resolution external to organizations
- ISO 10004:2018 Quality management — Customer satisfaction — Guidelines for monitoring and measuring
- RFC5424 - The Syslog Protocol (SYSLOG)

de facto Standards

- Oracle: Java logger API
- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala

Tools

- Tools: Logging Tools

[18] Techopedia, “Techopedia Data Logging,” 2 December 2017. https://www.techopedia.com/definition/596/data-logging

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:1_core:7_tools:1_log

Last update: **2021/05/10 17:20**

2.2.1.2.2 Semantic Web

return to Tools

The Semantic Web extends the World Wide Web (www) through standards developed by the World Wide Web Consortium (W3C).[19] These standards provide common data formats and exchange protocols on the web, typically using Resource Description Framework (RDF). RDF facilitates data merging even when underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all data consumers to be updated or modified.[20] "The Semantic Web provides a common framework allowing data sharing and reuse across applications, enterprises, and community boundaries".[21] The Semantic Web is therefore regarded as an integrator across different content, information applications, and systems.

Standards

Technical Standards

- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- OMG: Ontology Definition Metamodel (ODM)

de facto Standards

- None at this time

Tools

- None at this time

[19] "XML and Semantic Web W3C Standards Timeline" (PDF). 2012-02-04. http://www.dblab.ntua.gr/~bikakis/XML%20and%20Semantic%20Web%20W3C%20Standards%20Timeline-History.pdf

[20] Resource Description Framework, https://www.w3.org/RDF/

[21] "W3C Semantic Web Activity". World Wide Web Consortium (W3C). November 7, 2011. http://www.w3.org/2001/sw/

2.2.1.2.3 Open Source Communities

return to Tools

An important tool used within DIDOs is the use of Open Source Software (OSS). The use of OSS is not simply publishing software and allowing people to use it, but requires the establishment of an entire community with carefully developed rules and procedures to guide the design, development, release, maintenance, and sunsetting of the software.

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- Community tools

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:1_core:7_tools:1_oss

Last update: **2021/05/10 17:20**

2.2.1.3 Case Management

return to Fundamental Views

A major problem confronting DIDO Communities is the development of customer support to deal with issues encountered with either DIDO Transactions, Smart Contracts (especially those written externally by third parties), or DIDO OSS. For example, people have come to expect that when they use financial services, there will be recourse if transactions have unintended consequences. Recently, Coinbase, which provides an easy-to-use service for trading cryptocurrencies such as Bitcoin, Litecoin, and Ethereum, was hit with class action lawsuits alleging insider trading and also unlawful and unfair business practices.[4] In traditional banking, lack of recourse was part of the motivation behind the Dodd-Frank Wall Street Reform and Consumer Protection Act [5] , especially Section 1034 “Response to Consumer Complaints and Inquiries.” [6] Case Management must be applied to both of the organizational parts of DIDO Communities: software and fungible data (i.e. currency) management.

There are several categories of problems that can arise in a distributed system:

- Those involving distributed data
- Those involving the software used to distribute the data
- Those on a local node (machine)

Problems with Distributed Values (Domain Issues)

Problems that arise on the node network with the values stored on a node or set of nodes are generally domain issues. These cases generally have to do with the implementation of a DIDO Domain (i.e., Bitcoin cryptocurrency versus the Bitcoin platform). Therefore, the case is reported to the DIDO Domain. If the problem can be resolved at the domain level that's as far as the case needs to go. However, sometimes these cases need to be resolved at the domain and platform levels, thereby requiring two cases.

Problems with Distributed Software (Platform Issues)

Problems that arise on the node network having to do with conflicts in valid values stored on a node or set of nodes are generally platform issues. Generally, there should be no conflict with the values on any of the nodes since the DIDO implementations employ consensus methodologies, which form a large part of the added value of the individual DIDO platforms. For example, Bitcoin uses a Proof of Work (POW) methodology, whereas Ethereum and others use a Proof of State (PoS) methodology.

Problems with Node

Sometimes a node within the node network has problems. In a DIDO that has built-in redundancy, validation, and verification, this is generally not a problem and should be handled by the the original design. However, if nodes with a particular configuration (i.e., hardware, operating system, patches, security software, network cards, etc.) have issues, this could have consequences on the overall health of the domain.

Summary

Regardless of the location of the source for the case, most domains or platforms use Open Source Software and a bug tracking process based on a particular bug tracking tool such as Bugzilla, Jira, or Git-bug.

Generally, when consensus is reached for the correct value and requires a software upgrade, these cases are resolved using a Soft Fork. When consensus cannot be reached, a Hard Fork can occur within the domain.

Standards

Technical Standards

- ISO 10001:2018 Quality management — Customer satisfaction — Guidelines for codes of conduct for organizations
- ISO 10002:2018 Quality management — Customer satisfaction — Guidelines for complaints handling in organizations
- ISO 10003:2018 Quality management — Customer satisfaction — Guidelines for dispute resolution external to organizations
- ISO 10004:2018 Quality management — Customer satisfaction — Guidelines for monitoring and measuring
- OMG: Case Management Model and Notation (CMMN)
- OMG: Test Information Interchange Format (TestIF)

de facto Standards

- Bitcoin: Bitcoin Improvement Proposals (BIPs)
- Ethereum: Ethereum Improvement Proposals (EIPs)

Tools

- Tools: Bug and Issue Tracking

[4] S. Chang, “Coinbase Hit with 2 Class Action Lawsuits: Accused of Insider Bitcoin Cash Trading,” 4 March 2018. [Online]. https://www.investopedia.com/news/coinbase-hit-2-class-action-lawsuits-accused-insider-bitcoin-cash-tra

ding/

[5] Investopedia, “Dodd-Frank Wall Street Reform and Consumer Protection Act,” http://www.investopedia.com/terms/d/dodd-frank-financial-regulatory-reform-bill.asp.

[6] International Association of Risk and Compliance Professionals (IARCP), “Dodd Frank Act Text Section 1034,” 2010. http://www.dodd-frank-act.us/Dodd_Frank_Act_Text_Section_1034.html.

2.2.1.4 System of Systems (SoS)

return to Fundamental Views

To go beyond simply filling the roles traditionally fulfilled by cash, DIDOs (e.g., cryptocurrencies) need to become part of a fully distributed System-of-Systems (SoS) rather than a single monolithic product offering from a single source. Although the use of OSS helps mitigate the multiple source issue, it is not the panacea some envision because the specification of the system is by definition the current source code of the OSS.[25] The migration from a single monolithic product offering to an SoS means there should be multiple implementations available for most of the systems (or components) that comprise the DIDO Ecosystem (or Universe). Multiple DIDO implementations greatly reduce the risk of any part of the system being compromised by a single system, subsystem, or component failure, resulting in an even more robust network.

Furthermore, each component must be deterministic in its behavior, meaning that given the same set of inputs, the outputs will always be the same. In other words, not only will all the same implementations of a node produce the same output, but multiple implementations of a component within a node will provide the same results given the same inputs. In OSS systems, the OSS implementation **is** the reference implementation and provides the baseline definition of behavior resulting in a set of specific outputs for specific inputs. Selecting a component should be left to the individual participants in the network and treated as a business decision based on trust, mutual interests, and history. This means the ideal for all components is to have at least two different implementations, with each implementation acting to provide independent validation and verification of the other. This is not so different from current systems, where each node within the blockchain running the same code and getting the same results validates the transaction; differences in the results indicate a potentially compromised node.

For example, a successful DIDO could have worldwide usage (i.e., Bitcoin, Ethereum, etc.). With an SoS approach, each country in which a transaction is executed can have its own implementations of the various components and these implementations can reflect the rules and regulations on reporting and logging of the governing organization. For example, the Swiss might not want to have their transactions reported to the USA, China, Russia or even EU members owing to their unique privacy laws and Data Residency issues. [26] They would therefore select components that meet the needs of the Swiss rather than the world at large.

Technical Standards

- OMG: Systems Modeling Language (SysML)
- OMG: Unified Architecture Framework (UAF)
- ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes
- ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality

de facto Standards

- None at this time

[25] Section 2.2.1.2 covers the Open Source Paradigm in more detail.

[26] Object Management Group (OMG), “Data Residency Challenges and Opportunities for Standardization,” March 2017. [Online]. http://www.omg.org/cgi-bin/doc?mars/17-03-22.pdf.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:1_core:1_sos

Last update: **2021/05/10 17:22**

2.2.1.5 Quality

return to Fundamental Views

The ISO/IEC 25010 standard provides consistent terminology for "specifying, measuring and evaluating system and software product quality".[27] The Consortium for Information & Software Quality (CISQ) [28] provides the following diagram, highlighting eight ISO defined software quality characteristics and their associated sub-characteristics.[29] These ISO defined characteristics must be applied to both DIDO software and coinage organizations (e.g., Bitcoin, Ethereum, IOTA), especially Reliability, Performance Efficiency, Security, and Maintainability, since these are candidates for automation.

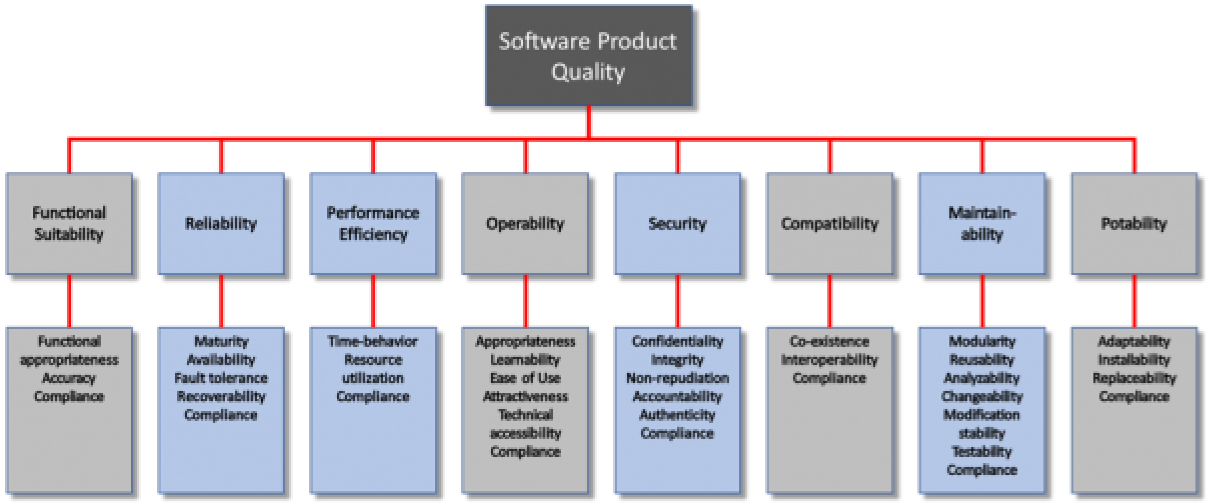

Software Product Quality
Functional Suitability
Reliability
Performance Efficiency
Operability
Security
Compatibility
Maintain-ability
Potability
Functional appropriateness Accuracy Compliance
Maturity Availability Fault tolerance Recoverability Compliance
Time-behavior Resource utilization Compliance
Appropriateness Learnability Ease of Use Attractiveness Technical accessibility Compliance
Confidentiality Integrity Non-repudiation Accountability Authenticity Compliance
Co-existence Interoperability Compliance
Modularity Reusability Analyzability Changeability Modification stability Testability Compliance
Adaptability Installability Replaceability Compliance

Figure 10: Quality Characteristics and Measures Specifications

Management

Management is part of SQuaRE and defines all common models, terms, and definitions referenced by all other standards from the SQuaRE series.

- ISO/IEC/IEEE 25000:2014 SQuaRE -- Guide to SQuaRE
- ISO/IEC 25001:2014 SQuaRE -- Planning and Management
- ISO/IEC/IEEE 90003:2018 Software engineering – Guidelines for the application of ISO 9001:2015 to computer software
- ISO 9001:2015 Quality management

Modelling

Modelling is part of SQuaRE and provides detailed quality models for computer systems and software products, quality in use, and data.

- ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models
- ISO/IEC 25012:2008 SQuaRE -- Data Quality Model

Measurement

Measurement is part of SQuaRE and includes a software product quality measurement reference model, mathematical definitions of quality measures, and practical guidance for their application.

- ISO/IEC 25020:2007 SQuaRE -- Measurement Reference Model and Guide
- ISO/IEC 25021:2012 SQuaRE -- Quality Measure Elements
- ISO/IEC 25022:2016 SQuaRE -- Measurement of Quality in Use
- ISO/IEC 25023:2016 SQuaRE -- Measurement of System and Software Product Quality
- ISO/IEC 25024:2015 SQuaRE -- Measurement of Data Quality

Requirements

Requirements are part of SQuaRE and help specify quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a software product to be developed, or as input for an evaluation process.

- ISO/IEC 25030:2007 SQuaRE -- Quality Requirements

Evaluation

Evaluation is part of SQuaRE and provides requirements, recommendations, and guidelines for software product evaluation.

- ISO/IEC 25040:2011 SQuaRE -- Evaluation Process
- ISO/IEC 25041:2012 SQuaRE -- Evaluation Guide for Developers, Acquirers and Independent Evaluators
- ISO/IEC 25045:2010 SQuaRE -- Evaluation Module for Recoverability
- OMG: Test Information Interchange Format (TestIF)

[27] International Association of Risk and Compliance Professionals (IARCP), “Dodd Frank Act Text Section 1034,” 2010. http://www.dodd-frank-act.us/Dodd_Frank_Act_Text_Section_1034.html.

[28] International Standards Organization (ISO), “The ISO/IEC 25000 series of standards,” http://iso25000.com/index.php/en/iso-25000-standards?limit=4&start=4.

[29] Consortium for Information & Software Quality (CISQ) http://it-cisq.org/.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:1_core:5_qual

Last update: **2021/06/16 11:49**

2.2.1.6 Open Source Paradigm

return to Fundamental Views

Using a DIDO is not just a simple shift in policies, procedures, and practices. It is a change in the entire architectural paradigm. Moving away from centralized control to distributed requires a complete change in how the system is normalized into systems, subsystems, components, etc. It also requires a shift in the basic underlying principles of the system. DIDOs are generally:

- Comprised of thousands if not millions of independent nodes
- Outside the control of any one individual or corporation
- Lacking any centralized authority; decisions are made by consensus

The DIDO architecture does not represent a single unified enterprise, but rather a loosely defined confederation of domains that requires systems integration (SI)[30]. Although SI is not new to enterprises, the granularity and types of components require a rethink. Within the DIDO environment, the definition of a platform shifts from hardware, operating system, software languages, and services (e.g., web, app, database) components to the DIDO Platform components. It is the responsibility of the DIDO Platform to isolate the enterprise from traditional platform concerns.

The granularity of the data elements within an enterprise can also shift to smaller, more isolated objects, which represent only a portion of the traditional Data Model (DM). In other words, the enterprise's data model is not going to be deployed into a single DIDO, nor should it. Enterprise data stores will continue to be needed but will be augmented and complemented by the DIDO. Some data will reside completely within the enterprise data stores, some data will reside completely within the DIDO, and some data will straddle both. Data that straddles both will require the definition of policies and procedures to ensure their data integrity.

Relevant Open Source Standards

The cultural shift from a stove-piped corporate or enterprise culture with almost complete control, to being a systems integrator participating in numerous distributed communities covering a wide range of domains, requires committed leadership and concerted effort by all the players.

Technical Standards

- None at this time.

de facto Standards

- There are none at this time but the Open Source Communities often rely heavily on the products of both Technical Standards Bodies and de facto Standards Bodies in building their projects.

- There are many guides available for participating in Open Source initiatives. **Talk Openly Develop Openly** (TODO) provides an extensive reading list.[31]. TODO also provides the following excellent guide as a place to start: TODO: Participating in open source communities.

[30] **System Integrator** - An individual or organization that builds systems from a variety of diverse components. With increasing complexity of technology, more customers want complete solutions to information problems, requiring hardware, software and networking expertise in a multi-vendor environment. https://www.pcmag.com/encyclopedia/term/52450/systems-integrator

[31] TODO Open Source Reading List, https://todogroup.org/guides/open-source-reading-list/

2.2.1.7 Assurance

return to Fundamental Views

The existing strategy for software and system assurance is already defined by the Systems and software Quality Requirements and Evaluation (SQuaRE). It establishes a common framework for analysis and exchange of information related to system assurance and trustworthiness, and defines the following kinds of assurance that need to be addressed: Information Assurance (IA), Safety Assurance (SfA), Software Assurance (SwA), Mission Assurance (MA) and System Assurance (SysA).

Assurance does not yield binary true / false answers. Assurance is a measure of risk which is a probability or threat of damage, injury, liability, loss, or any other negative occurrence that is caused by external or internal vulnerabilities, and that may be avoided through preemptive action.[32]. Assurance is best handled using Structured Assurance Case Metamodels (SACMs) for each of the assurances detailed above. A DIDO community of interest (CoI) best interest is to provide assurance measurements of their software, especially those CoIs that are offering “coinage” products to provide formal SACM results.

[32] Business Dictionary, Accessed 1 June 2020, http://www.businessdictionary.com/definition/risk.html

2.2.2 Node Network View

return to Technical Views

There are slightly different variations for the definitions of a Node Network, Node, Full Node, Light Node, and Miner Node depending on the DIDO Platform.[33],[34],[35],[36]

- The **Node Network** is a collection of interconnected computers communicating over a network of equally privileged computers (i.e., there are no central authoritative computers). The node network is sometimes referred to as a **peer-to-peer (P2P) network**.
- A **Node** is an individual computer that participates as an equal within the node network. A node is sometimes referred to as a **peer**. Nodes receive input, perform one or more operations on those inputs, and returns an output.

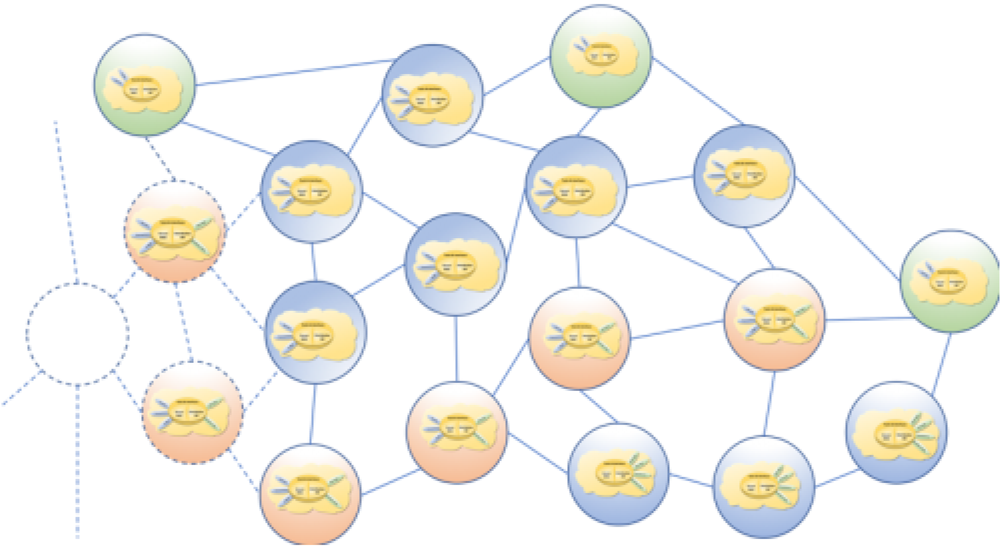


Figure 11: Node Network of Nodes.

- 2.2.2.1 Network View
- 2.2.2.2 Node View
- 2.2.2.3 Node Architecture
- 2.2.2.4 Messaging View

[33] “What is a Bitcoin node?”, Nate Eldredge, 14, December 2013, https://bitcoin.stackexchange.com/questions/18736/what-is-a-bitcoin-node

[34] “What are Ethereum Nodes And Sharding?”, Ameer Rosic, 2017, https://blockgeeks.com/guides/what-are-ethereum-nodes-and-sharding/

[35] “IOTA Full Node—That’s Why a Full Node Is Important for IOTA!”, Marko Vidrih, February 2019, https://medium.com/altcoin-magazine/iota-full-node-thats-why-a-full-node-is-important-for-iota-1c46280d7712

[36] “Introduction to IPFS: Run Nodes on Your Network, with HTTP Gateways”, Ross Bulat, 3 December 2018, https://medium.com/@rossbulat/introduction-to-ipfs-set-up-nodes-on-your-network-with-http-gateways-10e21ea689a4

2.2.2.1 Network View

return to Node Network View

The Network View looks at the DIDO as a single entity. Even though the network is comprised of a collection of individual nodes, individual nodes work as one because they:

- Interact as a peer in community of peers (i.e., Peer to Peer (P2P))
- Use a single Data Object (i.e., base coinage)
- Process transactions according to the rules of the community

For example, there is a system of nodes involved in the lifecycle of Bitcoins. However, it makes no sense for public records such as births, deaths, marriages, and divorces to be in the Bitcoin system of nodes. Thus, the public records could form their own system of nodes using the Bitcoin software but restrict this network to storage of pubic records.

Obviously, Bitcoin's main purpose is to transfer assets around the world at high speed and with low overhead. However, these are not the capabilities motivating the use of DIDO networks for low volatility public records such as those which reflect the natural rates of births, deaths, marriages, and divorces. Such applications are usually under the jurisdiction of a single country or countries that have reciprocity agreements or treaties.

Similarly, the need to establish a private system of nodes might exist for internal use only users (e.g., government enclaves, large corporations) that have enough distributed resources to support the network. Although the current cryptographic protocols provide "security" to a DIDO, with the advent of quantum computing [37], a reliance on these algorithms in the future for highly classified or private data may not be acceptable. These risks provide even more justification for the development of private DIDO networks.

As a general rule, the larger the system of nodes, the more secure and tamper-proof the data held within the network becomes, which means that for networks to be viable from a security perspective, the number of nodes in the collection might have a minimum.

- 2.2.2.1.1 Secure Messaging
- 2.2.2.1.2 Transport
- 2.2.2.1.3 Security
- 2.2.2.1.4 Protocol
- 2.2.2.1.5 Distribution Software

[37] MIT Technology Review, "Quantum Computers Pose Imminent Threat to Bitcoin Security," 8 November 2017. https://www.technologyreview.com/s/609408/quantum-computers-pose-imminent-threat-to-bitcoin-security/

2.2.2.1.1 Secure Messaging

return to Network View

The lowest functional layer of a node component is secure messaging. It is essential that a distributed system be able to have trusted, reliable, tamper resistant, and anti-snoop data flow between nodes in the network. Secure messaging must also include a standardized, consistent, and predictable way to marshal data between nodes. For example, the Endianness of each node may be different. This requires a solid transport mechanism, an excellent security infrastructure, and a standardized way to distribute data quickly and efficiently.

Standards

Technical Standards

- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC6749 - The OAuth 2.0 Authorization Framework
- RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC2315 - Cryptographic Message Syntax
- RFC3447 - PKCS #1: RSA Cryptography Specifications
- RFC7061 - eXtensible Access Control Markup Language (XACML) XML Media Type
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2104 - Keyed-Hashing for Message Authentication (HMAC)
- OMG: Data Distribution Service (DDS)
- OMG: DDS Security (DDS-SECURITY)

de facto Standards

- ZeroMQ Distributed Messaging
- Google: gRPC
- Google: Protocol Buffers
- Linux Foundation: Open Middleware Agnostic Messaging API (OpenMAMA)
- Linux Foundation: Open Messaging
- BIP 0070 - Payment Protocol

2.2.2.1.2 Transport

return to Network View

A cornerstone of DIDO networks is the transport component, enabling nodes to communicate with a very high degree of trust and reliability. Although it is possible to create a network of nodes that do not use Internet Protocol (IP), the breadth and acceptance of such an endeavor is questionable.

> *DIDO relies on the transport layer as defined by the Open Systems Interconnection (OSI) model. The transport layer is the layer in the open system interconnection (OSI) model responsible for end-to-end communication over a network. It provides logical communication between application processes running on different hosts within a layered architecture of protocols and other network components. The transport layer is also responsible for the management of error correction, providing quality and reliability to the end user. This layer enables the host to send and receive error corrected data, packets or messages over a network and is the network component that allows multiplexing.*[3].

Standards

Technical Standards

- RFC0147 - The Definition of a Socket
- RFC0791 - Internet Protocol (IPv4)
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2246 - The TLS Protocol
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)

de facto Standards

- Google: Protocol Buffers
- ZeroMQ Message Transport Protocol (ZMTP)

Tools

- None at this time

[3] Techopedia, “Transport Layer,” 30 November 2017.https://www.techopedia.com/definition/9760/transport-layer

2.2.2.1.3 Security

return to Network View

A DIDO is focused on two elements of security:

Internet Security: intended to protect the data flowing through the network. A major part of the functionality of a DIDO is its distributed nature over the internet and trust that it processes valid and accurate transactions.

> *Internet Security is essential in establishing trust to its success. Internet security is a catch-all term for a very broad issue covering security for transactions made over the Internet. Generally, Internet security encompasses browser security, the security of data entered through a Web form, and overall authentication and protection of data sent via Internet Protocol.*[39]

Information Security: intended to protect the confidentiality, integrity, and availability of the information contained within the DIDO network.

> *Information security (IS) is designed to protect the confidentiality, integrity and availability of computer system data from those with malicious intentions. Confidentiality, integrity and availability are sometimes referred to as the CIA Triad of information security. This triad has evolved into what is commonly termed the Parkerian hexad, which includes confidentiality, possession (or control), integrity, authenticity, availability and utility.*[40]

Standards

Technical Standards

- OMG: DDS Security (DDS-SECURITY)

de facto Standards

- None at this time

Tools

- None at this time

[39] Techopedia, “Techopedia Internet Security,” 30 November 2017. https://www.techopedia.com/definition/23548/internet-security

[40] Techopedia, “Information Security (IS),” 30 November 2017. https://www.techopedia.com/definition/10282/information-security-is.

2.2.2.1.4 Protocol

return to Network View

At the most elementary level, a communication protocol is comprised of a set of rules and guidelines that allow two or more nodes to communicate successfully over a network. DIDO networks must have a standardized robust protocol as the basis for processing transactions and to provide other communications between nodes.

> *A protocol is a set of rules and guidelines for communicating data. Rules are defined for each step and process during communication between two or more computers. Networks have to follow these rules to successfully transmit data.*[41)]

Standards

Technical Standards

- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC0791 - Internet Protocol (IPv4)
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- RFC0768 - User Datagram Protocol (UDP)
- RFC1112 - Host Extensions for IP Multicasting
- RFC3339 - Date and Time on the Internet: Timestamps
- RFC8259 - The JavaScript Object Notation (JSON) Data Interchange Format
- OMG: Data Distribution Service (DDS)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Web-Enabled DDS (DDS-WEB)

de facto Standards

- Bitcoin: Bitcoinj Developer's Documentation
- EIP 1474: Remote Procedure Call (RPC) specification (DRAFT)
- EIP 234: `blockHash` to JSON-RPC filter options (DRAFT)
- EIP 1898: ERC-NN Add `blockHash` to JSON-RPC methods which accept a default block parameter (DRAFT)
- EIP 1898: ERC-NN Add `blockHash` to JSON-RPC methods which accept a default block parameter (DRAFT)
- EIP 1193: Ethereum Provider JavaScript API (DRAFT)
- Ethereum: cpp Project

- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- Google: gRPC

Tools

- None at this time

[41] Techopedia, “Technopedia Protocol,” 30 November 2017. https://www.techopedia.com/definition/4528/protocol.

2.2.2.1.5 Distribution Software

return to Network View

Each node has a distribution software component. It is responsible for distributing and coordinating data and software throughout a DIDO network. Probably one of the best-known implementations of distribution software is the Domain Name System (DNS), which forms the backbone of the Internet:

> *Domain name system (DNS) is a hierarchical naming system built on a distributed database. This system transforms domain names to IP addresses and makes it possible to assign domain names to groups of Internet resources and users, regardless of the entities' physical location.*[42]

However, with the advent of the Blockchain papers by Satoshi Nakamoto [43] another major player, Bitcoin, has emerged in the distribution software space.

Standards

Technical Standards

- RFC1034 - Domain Names - Concepts and Facilities
- RFC1035 - Domain Names - Implementation and Specification
- RFC3596 - DNS Extension to support IP Version 6
- RFC5011 - Automated Updates of DNS Security (DNSSEC) Trust Anchors
- RFC6376 - DomainKeys Identified Mail (DKIM) Signatures
- RFC6891 - Extension Mechanisms for DNS (EDNS(0))

de facto Standards

- None at this time

Tools

- None at this time

[42] Techopedia, “Techopedia Domain Name Service (DNS),” 30 November 2017. https://www.techopedia.com/definition/24201/domain-name-system-dns.

[43] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. https://bitcoin.org/bitcoin.pdf.

2.2.2.2 Node View

return to Node Network View

The Node View represents the internals of any particular node within the node network. As represented in the figure below, it is comprised of five layers; however, the layers are not set in stone – each implementation of a node within a domain may be organized differently.

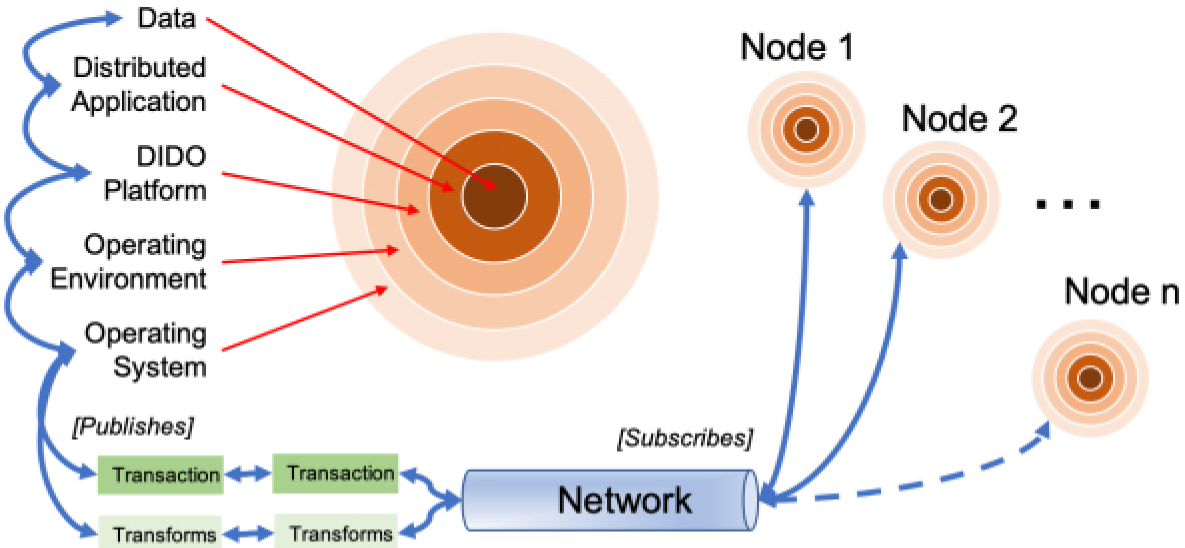


Figure 12: The idealized five layers of a Node.

At the boundary between each layer there is an interstitial layer formed by an Application Programming Interface (API), which is usually defined by one or more technical or de facto standards.

As each transaction, transform, or stream of data flows to or from a node over the network, it must travel through and/or interact with the:

- 2.2.2.2.1 Operating System (OS)
- 2.2.2.2.2 Operating Environment
- 2.2.2.2.3 DIDO Platform
- 2.2.2.2.4 Distributed Applications

2.2.2.2.1 Operating System (OS)

return to Node View

An operating system (OS), in its most general sense, is software that allows a user to run other applications in a computing device, as well as Virtual Machine applications, which emulate another computer. While it is possible for a software application to interface directly with hardware, it is not advisable from a portability or lifecycle perspective. Software applications that access hardware resources or other computer components directly pose a security risk.

Operating systems provide a common, well documented, and tested set of libraries, which abstract the idiosyncrasies of the host computer away from its applications.

An OS primarily manages a computer's hardware resources, including:

- Input devices such as a keyboard, mouse, track pad, touch screens, camera, microphone, scanners, or sensors
- Output devices such as display monitors, speakers, printers, or faxes
- Network devices such as modems, router, wired and wireless Internet Protocol network connections, and Bluetooth
- Storage devices such as internal and external disks
- Memory devices

The OS also manages a computer's:

- CPU
- Processes
- Privileges
- Cache
- Energy, i.e., power management

Standards

Technical Standards

- IEEE 1003.1-2017 - IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications (**NOTE:** See UNIX)
- ISO/IEC 23360-1:2006 Linux Standard Base (LSB) core specification 3.1 -- Part 1: Generic specification
- ISO/IEC The Linux Standard Base 5 Specification Series (LSB 5)

de facto Standards

- Apple: Darwin

- Apple: iOS
- Apple: MacOS
- Google: Android
- Microsoft: Windows API

Tools

- None at this time

2.2.2.2.2 Operating Environment

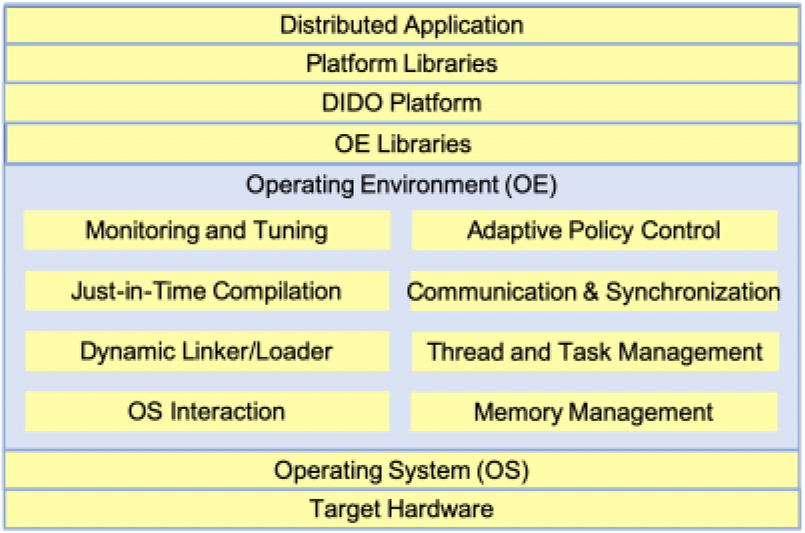
return to Node View

The concept of an Operating Environment (Run Time System) is generally traced back to *Ada*, which defined: an abstract interface to the underlying operating system, thread and task management, as well as mechanisms for monitoring, tuning, and performing dynamic memory management (all to protect against access to unallocated memory, buffer overflow errors, range violations, off-by-one errors, array access errors, and other detectable poor coding practices)[8].

Another major advancement came with Java and the Java Virtual Machine (JVM), which added a Just-In-Time (JIT) compiler, virtual processor, and an interpreter[9].

Microsoft later introduced the .NET framework, which runs on the Windows operating system. .Net includes Web Services, Web Forms, and Windows Forms in the Operating Environment[10]. Subsequently, Mono was introduced and is now sponsored by Microsoft in order to allow cross-operating system development and deployment of .Net applications. The Mono Framework is based on ECMA Standards for C# and the Common Language Runtime[11];

Figure 2: Generalized Operating Environment (OE) components[12]



Standards

Technical Standards

- ISO/IEC 9899:2018 Programming languages -- C
- ISO/IEC 14882:2017 Programming languages -- C++

- ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)
- ECMA: Standard ECMA-334 - C# Language Specification
- ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)

- ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File
- ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics
- RFC5424 - The Syslog Protocol (SYSLOG)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- W3C: HTML5 (HTML5)
- W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)
- W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- W3C: XSL Transformations (XSLT) Version 3.0
- W3C: Document Object Model (DOM) Level 3 Core Specification
- W3C: XML Path Language (XPath) 3.1
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)
- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

de facto Standards

- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala
- Bitcoin: Developer's Guidance
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- EIP 20: ERC-20 Token Standard
- Ethereum: Ethereum Virtual Machine (EVM)

- Ethereum: Solidity Language Specification
- Linux Foundation: Hyperledger
- Oracle: The Java® Language Specification SE 8 Edition
- Oracle: The Java® Virtual Machine Specification JVM
- Oracle: Java logger API
- Google: Go (software language)
- InterPlanetary File System (IPFS)

Tools

- None at this time

[8] https://en.wikipedia.org/wiki/Ada_(programming_language

[9] “Understanding the JVM Architecture”, Joydip Kanjilal, Developer.com, 16 February 2015, https://www.developer.com/java/data/understanding-the-jvm-architecture.html

[10] “Components of .Net Framework”, DeveloperIn.Net, Jayanthan JVP, http://www.developerin.net/a/39-Intro-to-.Net-FrameWork/23-Components-of-.Net-Framework

[11] “Cross platform, open source .Net Framework”, The Mono Project, https://www.mono-project.com/

[12] “Language Run-Time Systems: An Overview”, Evgenij Belikov, Heriot-Watt University, School of Mathematical and Computer Sciences Riccarton, EH14 4AS, Edinburgh, Scotland, UK https://pdfs.semanticscholar.org/b20c/0295a0661a4c175fcd5acc0ea49e9caf3ca1.pdf

2.2.2.2.3 DIDO Platform

return to Node View

A DIDO platform definition encapsulates the complete environment that supports a running DIDO. The DIDO platform covers the hardware, operating system (OS), DIDO software, and the connection to the network. There are potentially many related DIDO platforms defined for any particular domain. Each DIDO platform can vary the hardware, OS, type of network connection, and potentially the DIDO software if the software can interoperate.

For example, a domain defined for a supply chain might have a set of DIDO platforms defined. A DIDO platform for Windows, Linux, UNIX, Android, Mac OS, and IOS can work on TCP/IP machines or TCP/UDP protocols.

Standards

Technical Standards

- None at this time

de facto Standards

- Bitcoin: Bitcoinj Developer's Documentation
- Bitcoin: Developer's Guidance
- Bitcoin: Bitcoin Improvement Proposals (BIPs)
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- Google: Protocol Buffers

Tools

- None at this time

2.2.2.2.4 Distributed Applications

return to Node View

A **distributed application** is software that does not reside on a single computer, centralized server, decentralized server, or set of clustered servers. The application is distributed among nodes on the network. Each instance of the application on each node executes the same code and gets the same results based on the current state of the data and the input data sent to it. In other words, the distributed application's nodes are deterministic in nature.

Standards

Technical Standards

- RFC6455 - The WebSocket Protocol
- RFC0793 - Transmission Control Protocol
- RFC2104 - Keyed-Hashing for Message Authentication (HMAC)
- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC0791 - Internet Protocol (IPv4)
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- RFC6749 - The OAuth 2.0 Authorization Framework
- RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC1112 - Host Extensions for IP Multicasting
- RFC2315 - Cryptographic Message Syntax
- RFC3447 - PKCS #1: RSA Cryptography Specifications
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2246 - The TLS Protocol
- RFC0768 - User Datagram Protocol (UDP)
- ISO/IEC 9899:2018 Programming languages -- C
- ISO/IEC 14882:2017 Programming languages -- C++
- ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)
- ECMA: Standard ECMA-334 - C# Language Specification
- ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)
- ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File
- ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics
- RFC5424 - The Syslog Protocol (SYSLOG)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification

- W3C: HTML5 (HTML5)
- W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)
- W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- W3C: XSL Transformations (XSLT) Version 3.0
- W3C: Document Object Model (DOM) Level 3 Core Specification
- W3C: XML Path Language (XPath) 3.1
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)
- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

de facto Standards

- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala
- Bitcoin: Developer's Guidance
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- EIP 20: ERC-20 Token Standard
- Ethereum: Ethereum Virtual Machine (EVM)
- Ethereum: Solidity Language Specification
- Linux Foundation: Hyperledger
- Oracle: The Java® Language Specification SE 8 Edition
- Oracle: The Java® Virtual Machine Specification JVM
- Oracle: Java logger API
- Google: Go (software language)
- InterPlanetary File System (IPFS)

Tools

- Tools: Network Traffic Analysis

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:2_node:4_dapp

Last update: **2021/03/29 10:43**

2.2.2.3 Node Architecture

return to Node Network View

The DIDO node architecture establishes a component model as a reference for evaluating the functionality or data available within a DIDO implementation. The reference components are conceptual in nature; thus they may or may not represent actual components within a DIDO implementation. However, the functionality of these reference components should be part of the implementations.

At the most elementary level, reference components provide a common vocabulary for DIDO implementations. For example, Bitcoin does not have a separate standalone component that performs secure messaging; however, the Bitcoin software *does* provide secure messaging functionality using a protocol built upon cryptography. This binds the secure messaging and protocol together. It is possible to write applications that adhere to this protocol using the cryptographic rules and standards defined by Bitcoin and not use the Bitcoin software; however, the usual approach is to build upon the open source Bitcoin software and use it "as-is." Another example is where Bitcoin provides an actual component referred to as a Wallet, bearing in mind there are numerous other wallets available that can contain and manage Bitcoins.[49)]

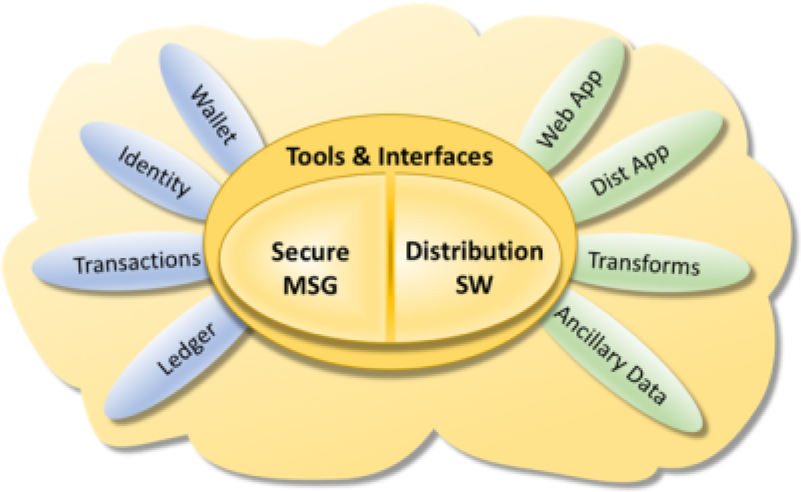


Figure 14: The DIDO Node Component Model

The DIDO node component model presented in this diagram (Figure 14) may look like a traditional stack or layered architecture as described in centralized or decentralized models; however, a very important difference is that the components or a subset of the components of this architecture are usually repeated at every DIDO node that participates in the DIDO network. At a minimum, the secure message and the distribution software components must exist at each node in order for the network to remain distributed and operational. In addition to providing for the core components required to securely communicate, a DIDO node can take on different roles or functions (refer to Figure 15). Some nodes may use all the components within the DIDO node component model while others may only use a subset of these components. For example, a smart contract node may use the identity, transaction, ledger, distributed app, and ancillary data components. In contrast, a Wallet node may just use wallet and identity components.

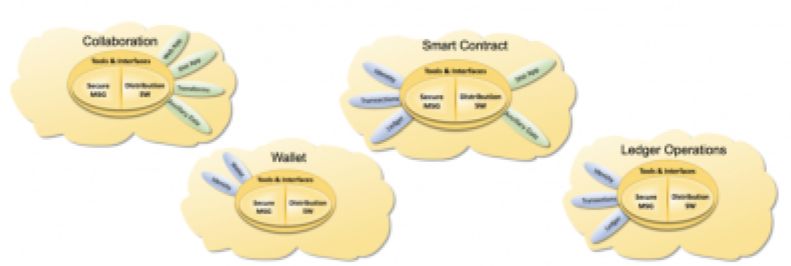


Figure 15: Examples of kinds of DIDO Nodes

When the entire DIDO network is assembled, it may look something like Figure 16. Each node has the basic common core components and the set of components required for it to fulfill its specific responsibilities.

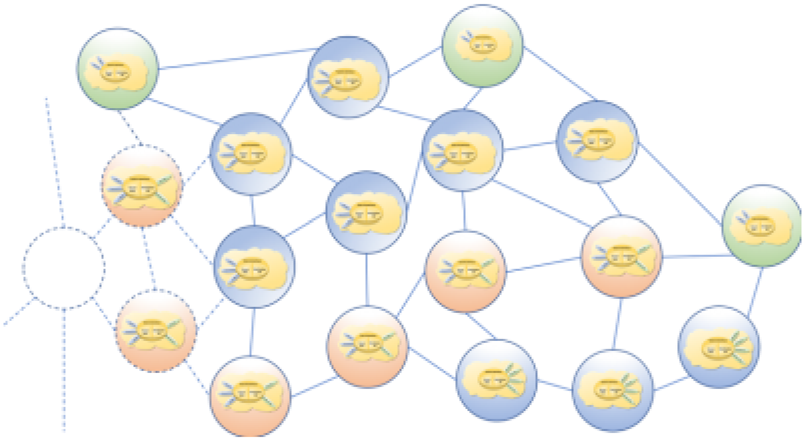


Figure 16: The DIDO Network Model

The DIDO network acts as a single system or entity, in which each node performs the operations required of it as a participant. For example, some DIDO networks may only provide a ledger, account numbers, and support transactions on the ledger. These would require each node to have the secure message, distribution software, ledger, identity and transaction components in common.

All the components within the network must be interoperable at the core functional level. For example, most of the nodes might run the Common Core version 1, whereas a subset might be running version 1.1. As long as the nodes within the network can interoperate and function together, the network is considered viable as a whole.

The interoperability of nodes within the DIDO and overall DIDO network viability become key benefits of DIDO implementations. This means that anything affecting interoperability is critical, and requires that interfaces and interactions (i.e., protocols) between the nodes must be the most conservative of all components. In other words, interfaces and interactions of the nodes are the mission critical aspects of the network and, therefore must be stable from the onset. Interruption or discontinuity in the critical path could result in a fork of the underlying ledger. An example of a fork is the “hard fork” in Ethereum called Byzantium:

> *The Byzantium hard fork is an update to ethereum’s blockchain that was implemented in October 2017 at block 4,370,000. It consisted of nine Ethereum Improvement Protocols (EIPs) designed to improve ethereum’s privacy, scalability and security attributes.*[50)]

Common Core

The Common Core contains components that are used by both the ledger and ancillary data subset of components, as well as characteristics and attributes that apply to all components within the DIDO network. There are three classes of common components: tools and interfaces, distribution software, and secure messaging. Common components can be used exclusively by one kind of DIDO node as defined in node architecture but can also span across the components within a node. For example, the transaction API is available to both the ledger and the ancillary data node and potentially by some of the tools. However, the transaction API may or may not be used as part of the ancillary data stack.

By definition, a ledger operation node must rely on the transaction API to access the ledger; however, a smart contract node may also require access to the ledger. In that case, its access is made exclusively through the transaction API. It is up to the implementation of a particular DIDO whether to access its ancillary data through a transaction API or a transform API. Conversely, a transaction might require ancillary data (i.e., monetary exchange rate, stock quotes, interest rates, market cap, or Beta, etc.) in order to complete. The Common Core contains an API that defines and allows for this access, sometimes referred to as an oracle.

Note: Transaction API and Transform API are defined in more detail in Section 2.2.2.4 Messaging View.

- 2.2.2.3.1 Immutable Data Objects
- 2.2.2.3.2 Ancillary Data
- 2.2.2.3.3 Semantic Web
- 2.2.2.3.4 Software

[49] A. Hertig, “Does the original Bitcoin Wallet Still Matter?,” 16 September 2016. https://www.coindesk.com/original-bitcoin-wallet-still-matter/.

[50] R. Sharma, “What is the Byzantium Hard Fork in Ethereum?,” 7 March 2018. https://www.investopedia.com/news/what-byzantium-hard-fork-ethereum/

2.2.2.3.1 Immutable Data Objects

return to Node Architecture

An immutable data object is data whose value cannot be changed. Any required update is recorded as a new immutable data object with a link back to its parent(s). An immutable data object represents things either real or virtual. The representation can range from a simple scalar value to a complex data structure of values. The things or items data objects represent are real-world things such as commodities or information. Commodities include things like gold, silver, grains, and so on. Information includes things like certificates of birth, death, marriage, or stock. Data objects can also represent virtual or abstract things such as virtual coins, frequent flier miles, loyalty points, data rights, etc.

The immutable data object component is focused on the care and maintenance of distributed, global data objects usually stored in a ledger, the identifiers (i.e., accounts) associated with the ledger, the transactions used to make updates to the ledger entries, and the wallets that contain identifier (i.e., account) information for a user. When immutable data objects represent a currency such as cryptocurrencies, the identifiers represent accounts; however, when the immutable data objects represent data such as commodities, inventories, and public records, the identifiers may not represent accounts but unique identifiers of the thing.

- 2.2.2.3.1.1 Ledger
- 2.2.2.3.1.2 Transactions
- 2.2.2.3.1.3 Identities
- 2.2.2.3.1.4 Wallets

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:3_nodearch:2_ido

Last update: **2021/03/29 10:44**

2.2.2.3.1.1 Ledger

return to Immutable Data Objects

Typically, an immutable data object is stored as a ledger. The term *ledger* is a distributed object that has its root within accounting systems and is comprised of a series of entries that represent an account at different stages in its life. This includes its value, an operation to be performed on the value, and potentially another account that receives or sends an amount. Ledger entries are immutable, i.e., once an entry has been added to the ledger, it can never be modified. New versions of the entry exist, and transactions capture the changes required to move between the data entry values.

For example, if the original value for an entry was AAAA and a transaction wants to modify the value to BBBB, the original value remains in the ledger and a new entry is made with the value BBBB. The new entry points back to the previous value that contained AAAA. Thus, by following the chain of modifications backwards, the provenance and ultimately the pedigree of the value can be determined.

The ledger is not limited to account numbers but can have general identifiers which represent fungible data. For example, a library has books, but the books are referenced by an identifier, not an account. Patrons of the library may have identifiers which might look like accounts, but simply identify the patron.

The ledger exists at all nodes within the network. In the blockchain implementations of DIDO, all the values in the ledger for a particular entity will be identical when all the outstanding transactions contained within a block have been validated and verified and properly distributed throughout the network. There are alternatives to blocks and blockchains. For example, Hashgraphs which have been proven to also solve the Byzantine general problem asynchronously without having to use expensive Bitcoin PoW algorithms. [51)]

[51)] G. Kinglsy, “Hashgraph vs. Blockchain Is the end of Bitcoin and Ethereum near?,” coincodex, January 2018. https://coincodex.com/article/1151/hashgraph-vs-blockchain-is-the-end-of-bitcoin-and-ethereum-near/.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:3_nodearch:2_ido:1_ledg

Last update: **2021/03/29 10:44**

2.2.2.3.1.2 Transactions

return to Immutable Data Objects

Transactions contain cryptographically signed data required to describe the creation, transfer, or destruction of fungible data representing an asset stored within the ledger. The transaction captures the change in state of the contents of the ledger. Currently, each implementation of Blockchain defines its own unique concept of a transaction which depends on the kind of fungible data stored in the ledger.

Transactions represent operations that can be performed on fungible data represented in the ledger. In a financial ledger, the money associated with an account is the fungible data and it is represented by the balance associated with an account associated with an entry in the ledger. Money can only be added to or deducted from an account. Note: The actual money is not stored in the ledger, only a balance representing money is stored in the ledger. A slightly higher level concept would be the transfer money from one account to another (i.e., deduct from account nnn1 and add to account nnn2).

2.2.2.3.1.3 Identities

return to Immutable Data Objects

Associated with items in the ledger are identifiers which associate fungible data represented in the ledger with accounts or associated with transactions. In classic financial Ledgers, these identifiers are generally account numbers that contain a balance (a tally) of the fungible data associated with the account and the entry number of the row within the ledger. For example, account nnn1 has a balance of $50. Ledger Entry ''ttt2'' adds $25 to account nnn1.

The evolution of advanced symbologies has helped the securities industry grow, but the limitations and costs imposed by the closed systems have become more apparent as companies and institutions continue to integrate operations on a global scale. Proprietary symbology now stands as one of the most significant barriers to increased efficiency and innovation in an industry that sorely needs it. Moreover, the lack of common identifiers is a key roadblock to achieving the holy grail of Straight-through Processing (StP). [52)]

[52)] Object Management Group (OMG), “Financial Industry Global Identifier® (FIGI™), v1.0,” December 2015. http://www.omg.org/spec/FIGI/1.0/.

2.2.2.3.1.4 Wallets

return to Immutable Data Objects

Wallets are not quite like physical wallets carried around in pockets or handbags. They are repositories where the cryptographic keys associated with the fungible data managed by the ledger are stored. These keys are required to unlock access to the fungible data in the ledger. For example, within the Bitcoin Blockchain, there is an identifier associated with each Bitcoin. To use the Bitcoin, a public and a private key are required. The private keys are stored in the wallet.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:3_nodearch:2_ido:4_wall

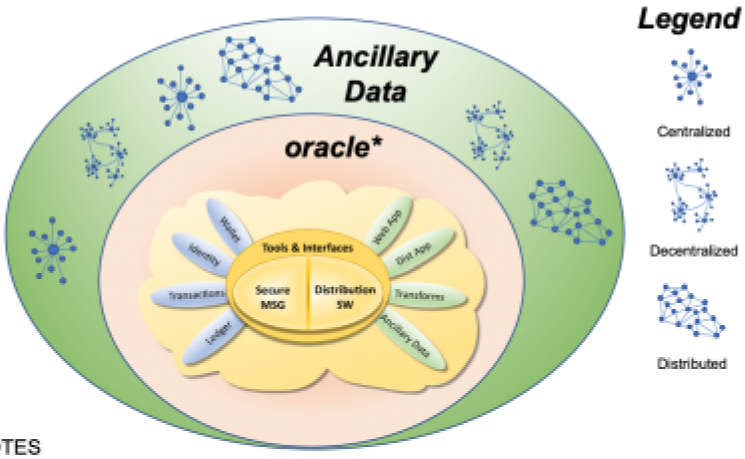
Last update: **2021/03/29 10:44**

2.2.2.3.2 Ancillary Data

return to Node Architecture

Ancillary ddta is focused on data that is not directly part of the fungible data stored in the ledger but is used in the support, care, and maintenance of the fungible data. Ancillary data is global in nature (all nodes need access to it) and is like the fungible data in the ledger, distributed in nature (all nodes have copies of the data). Some DIDO implementations may provide access to ancillary data through the use of oracles; however, if the ancillary data is not also implemented as a DIDO but as frontend to centralized or even decentralized data, many of the benefits of the distributed data are lost. For example, if the exchange rate between currencies is not also distributed but offered from a server through an oracle, access to the exchange data becomes a vulnerability to the use of the cryptocurrency data when an exchange rate is required.

Access to ancillary data depends on the source of the data. Some implementations will store all the ancillary data on the centralized server, others will provide access to ancillary data stored on a series of decentralized servers, while others will provide access to the data as fully decentralized networks (i.e., other DIDOs). Ancillary data is accessed through an oracle.



NOTES
* An oracle, in the context of blockchains and smart contracts, is an agent that finds and verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts. https://blockchainhub.net/blockchain-oracles/

Figure 17: Relationship of Ledger, Ancillary Data and Oracles to other Network Topographies

The ancillary data, when implemented as a DIDO, is hosted in its own network of nodes which may or may not be the same set of nodes as the fiduciary data. It is comprised of the ancillary data itself, a mechanism for transforming the data (i.e., not necessarily a ledger transaction), and software that is distributed on all the nodes in its network, called web applications, which may or not run on a node within the network.

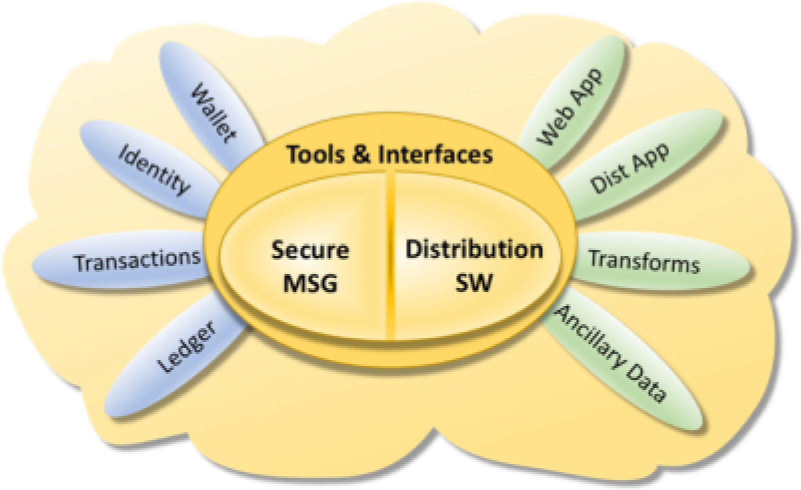


Figure 18: The Components in the Ancillary Data Stack of the DIDO Node

- 2.2.2.3.2.1 Journal
- 2.2.2.3.2.2 Transforms
- 2.2.2.3.2.3 Distributed Applications
- 2.2.2.3.2.4 Web Applications
- 2.2.2.3.2.5 Exchanges

2.2.2.3.2.1 Journal

return to Ancillary Data

The journal is data that is ancillary data to the main fungible data. It may or may not be implemented using a separate ledger and it may exist within another DIDO network, but its care and maintenance does not have to be part of the ledger’s transactions. For example, the monetary exchange rate between two currencies is ancillary to a cryptocurrency ledger; however, the almost instantaneous updates to the exchange rate would not be appropriate in the cryptocurrency ledger itself.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:3_nodearch:3_xdata:1_jrnl

Last update: **2021/03/29 10:45**

2.2.2.3.2.2 Transforms

return to Ancillary Data

There are different ways ancillary data can be transformed. If the ancillary data is implemented as another “blockchain” DIDO, then the ancillary data is stored within another, parallel ledger and then, naturally, the transforms would be accomplished using that ledger stack’s transactions.

However, another mechanism available to transform the ancillary data is operational transformation (OT). These are similar to transactions but there can be multiple changes made to the data at the same time by multiple parties. Ancillary data using operational transforms is not necessarily immutable.

2.2.2.3.2.3 Distributed Applications

return to Ancillary Data

One of the major extensions to the original Satoshi Nakamoto [1] [27] papers on peer-to-peer electronic cash systems is the use of oracles to access external data and Smart Contract to enforce business rules on ledger transactions. For example, an account holder wants to transfer Bitcoins from one account to another, but there is a "reserve" placed on the account to cover potential debts from option trading. This "reserve" information would not be part of the Ledger, but would be ancillary data. The enforcement of the "reserve" is done by a distributed application called a smart contract.

> *A distributed application is software that is executed or runs on multiple computers within a network. These applications interact in order to achieve a specific goal or task. Traditional applications relied on a single system to run them. Even in the client-server model, the application software had to run on either the [dido:public:ra:xapend:xapend.a_glossary:c:client]], or the server that the client was accessing. However, distributed applications run on both simultaneously.*
>
> *With distributed applications, if a node that is running a particular application goes down, another node can resume the task.* [13)]

A distributed application (DApp) is software that is executed or runs on multiple computers within a network simultaneously. The software is deterministic meaning that given the same inputs, they all produce the same outputs. One of the main benefits of DApps is they are extremely durable and hardened against a single point of failure. Therefore, to be distributed, deterministic and durable, any services that the DApp must interact with must also be distributed, deterministic, and durable. This makes Distributed Application (ĐApp or DApp) interaction with traditional client/server cloud services difficult. Naturally, cloud services applications such as Software as a Service (SaaS) and Data as a Service (DaaS) have some redundancy and reliability built into them; however, they cannot achieve the same level of robustness as a DApp. Therefore, SaaS and DaaS need to expose their functionality using a proxy DApp, which by its nature has latency. This latency could adversely affect the DApp's deterministic and durability nature.

Another important aspect of a DApp is that it should be runtime environment agnostic: allowing as many DIDO nodes into the DIDO network as possible. This can be achieved using Virtual Machine (VM) such as a Java Virtual Machine (JVM) or interpretive engines such as those available with ECMAScript. There are some platform specific virtual machines available such as the Common Language Runtime (CLR) which are not standardized and do not run with the deterministic rigor on non-Windows platforms. Consequently, the list of languages allowable in DApps is limited to those that have a standardized runtime environment (i.e., VM or Engine) that runs on multiple platforms.

[13)] Techopedia, "Techopedia Definitions," 1 December 2017. https://www.techopedia.com/definition/23971/distributed-application.

2.2.2.3.2.4 Web Applications

return to Ancillary Data

Web applications are the main interface between the human end-users of a DIDO and the rest of the DIDO. The web applications do not have the same requirements of determinism as defined in the DIDO and can be subject to far less stringent timing requirements. The equivalent of a web application in the ledger stack is a wallet.

> *A web application or “web app” is a software program that runs on a web server. Unlike traditional desktop applications, which are launched by your operating system, web apps must be accessed through a web browser.* [53]

=-
=-

[53] TechTerms, “TechTerms Definitions,” 1 December 2017. https://techterms.com/definition/web_application.

2.2.2.3.2.5 Exchanges

return to Ancillary Data

Investopedia defines an Exchange as:

An exchange is a marketplace in which securities, commodities, derivatives and other financial instruments are traded. The core function of an exchange is to ensure fair and orderly trading and the efficient dissemination of price information for any securities trading on that exchange. Exchanges give companies, governments and other groups a platform from which to sell securities to the investing public. [54]

Although this definition is meant to be specific to existing physical and electronic exchanges covering “securities, commodities, derivatives and other financial instruments” offered on the New York Stock Exchange (NYSE), Nasdaq, London Stock Exchange (LSE), and Tokyo Stock Exchange (TSE), it is also applicable when immutable data objects themselves become financial instruments and when they are applied to public records, certificates, or supply chains.

An exchange on the surface may seem to be a trivial concept and if this were so, the development of laws, rules, and regulations governing them would also be trivial. However, a quick look at the rules governing the NYSE alone is daunting [55] . These rules are not to torment or torture traders, or to limit trades; rather they are to “ensure fair and orderly trading”.

As the DIDO exchange concepts grow and mature, the need for, and the development of, rules and regulations to “ensure fair and orderly trading” must also adopt the rules and regulations expected from any exchange and evolve to handle the unique benefits, functions, capabilities of DIDOs.

Some of the types of trades that an exchange could handle are:

- **Market Order (MKT)** – A market order is an order to buy or sell a stock at the best available price. Generally, this type of order will be executed immediately. However, the price at which a market order will be executed is not guaranteed. It is important for investors to remember that the last-traded price is not necessarily the price at which a market order will be executed. In fast-moving markets, the price at which a market order will execute often deviates from the last-traded price or “real time” quote. [56]
- **Limit Orders (LMT)** – A limit order is an order to buy or sell a stock at a specific price or better. A buy limit order can only be executed at the limit price or lower, and a sell limit order can only be executed at the limit price or higher. A limit order is not guaranteed to execute. A limit order can only be filled if the stock’s market price reaches the limit price. While limit orders do not guarantee execution, they help ensure that an investor does not pay more than a pre-determined price for a stock. [57]
- **Stop Order (STP)** – A stop order, also referred to as a stop-loss order, is an order to buy or sell a stock once the price of the stock reaches a specified price, known as the stop price. When the stop price is reached, a stop order becomes a market order. A buy stop order is entered at a stop price above the current market price. Investors generally use a buy stop order to limit a loss or to protect

a profit on a stock that they have sold short. A sell stop order is entered at a stop price below the current market price. Investors generally use a sell stop order to limit a loss or to protect a profit on a stock that they own. [58)]

- **Stop Limit Order (STPLMT)** – A stop-limit order is an order to buy or sell a stock that combines the features of a stop order and a limit order. once the stop price is reached, a stop-limit order becomes a limit order that will be executed at a specified price (or better). The benefit of a stop-limit order is that the investor can control the price at which the order can be executed. [59)]
- Some of the types of trades that need to be considered during an Exchange are **Fill-Or-Kill (FOK)** – An FOK order is an order to buy or sell a stock that must be executed immediately in its entirety; otherwise, the entire order will be cancelled (i.e., no partial execution of the order is allowed). [60)]
- **All-Or-None (AON)** – An Aon order is an order to buy or sell a stock that must be executed in its entirety, or not executed at all. However, unlike the FoK orders, Aon orders that cannot be executed immediately remain active until they are executed or cancelled. [61)]
- **Market If Touched (MIT)** – A market-if-touched, or MIT, order is a conditional order that becomes a market order when a security reaches a specified price. When using a buy market-if-touched order, a broker will wait until the security falls to a certain level before purchasing the asset. A sell market-if-touched order will activate when the price of a security rises to the specified level. [62)]

This document provides a list of standards associated with each of the components within the DIDO Reference Architecture. In this section the descriptive text for each standard or specification is taken directly from the original standard or specification and is properly attributed with a reference to that standard or specification. This has been done in order to aid readers of the DIDO RA to be able to determine applicability and usefulness of the standard or specification to the particular instance of a DIDO they are working on.

Re-writing the descriptive text can result in a misunderstanding of the original intent of those standards and specifications. If you have ever been involved in the writing of a standard or a specification, you'll be familiar with the long discussions and debates on the selection of each word and the punctuation used. Therefore, the text in the standards listed below is duplicated as much as possible “intact”.

[54)] Investopedia, “Exchange,”. https://www.investopedia.com/terms/e/exchange.asp#ixzz5OGeT4QUj.

[55)] New York Stock Exchange, “NYSE Tools,” http://wallstreet.cch.com/NYSETools/PlatformViewer.asp?selectednode=chp%5F1%5F2%5F1%5F35&manual=%2Fnyse%2Frules%2Fnyse%2Drules%2F.

[56)] U.S. Securities and Exchange Commission, “Market Order,” https://www.sec.gov/fast-answers/answersmktordhtm.html.

[57)] U.S. Securities and Exchange Commission, “Limit Orders,” SEC, https://www.sec.gov/fast-answers/answerslimithtm.html.

[58)] U.S. Securities and Exchange Commission, “Stop Order,” SEC, https://www.sec.gov/fast-answers/answersstopordhtm.html.

[59)] U.S. Securities and Exchange Commission, SEC, https://www.sec.gov/fast-answers/answersstoplimhtm.html.

[60] U.S. Security and Exchange Commission, SEC, https://www.investor.gov/additional-resources/general-resources/glossary/fill-or-kill-order.

[61] U.S. Security and Exchange Commission, SEC https://www.investor.gov/additional-resources/general-resources/glossary/all-or-none-order.

[62] Investopedia, Market If Touched - MIT, https://www.investopedia.com/terms/m/marketiftouched.asp#ixzz5OHOMW2ba]].

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:3_nodearch:3_xdata:4_xcgng

Last update: **2021/03/29 10:45**

2.2.2.3.3 Semantic Web

return to Node Architecture

A major promise of DIDO is the ability to use machines to automatically process data and transactions rather than to rely on humans “in the middle.” A key to automating many of these processes is realized through the adoption of the semantic web. In essence, the semantic web is the institutional memory and experience captured in machine readable form and available at each node. This eliminates the centralized human-centric requirements of the past and supports a distributed, automated solution. The semantic web uses formal semantics that unambiguously capture the vocabulary and ontology of the domain.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:3_nodearch:3_web

Last update: **2021/03/29 10:45**

2.2.2.3.4 Software

return to Node Architecture

By their very nature, DIDOs are run on distributed, decentralized computers that require software to operate. The DIDO software itself can be classified as ancillary data and therefore can be distributed just like data to each DIDO node. The DIDO software could include Smart Contract, Distributed Application (ĐApp or DApp), scripts, containers, and DIDO command line interface (CLI) commands.

Smart contracts are programs actually stored on the blockchain, and “triggered” to execute when a set of conditions are met. dApps are applications that don't reside on the blockchain but interact with the blockchain. dApps are used to communicate with smart contacts, and consequently with blockchain.

In other words, dApps can be considered as “blockchain-enabled” application, and smart contracts provide easy access the blockchain. This division of responsibilities is similar to the traditional separation used in Web Application (Web App) between the user-front-end (i.e., Client) and the backed (i.e., Server). Contract development is concerned with managing agreements or transactions.

There are many non-functional benefits of dApps such as:

- Maintainability
- Portability
- Scalability
- Interoperability

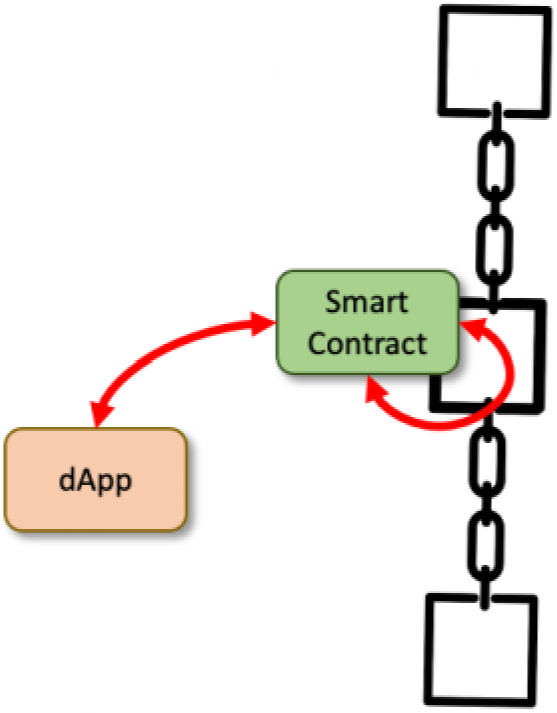


Figure 19: The Relationship Between the dApp, the Smart Contract and the blockchain

- **Note:** Not only is the data deployed on the blockchain, the smart contract itself is deployed on the blockchain. Therefore, before the Smart Contract can be used, it must be deployed to the blockchain. With slight modification, the dApp could interact with any system regardless of it is a distributed system or not as long as the interface remains the same.

2.2.2.4 Messaging View

return to Node Network View

There are three classes of messages that are used within the DIDO node network.

Transactions

Transactions are messages containing instructions on how to change the ledger from one known state to the next state. Transactions are sent to all the nodes within the Node Network providing instructions on how to modify the ledger data to the next state. When all the transactions have been applied to all ledgers on all of the individual nodes within the node network, the ledgers will have the same state and can be treated as a single datastore. These transactions are executed using a transaction Application Programming Interface (API), an API.

Transforms

Like transactions, transforms are instructions that change the state of a “ledger” from one state to another. Transforms are not as order dependent as transactions and are NOT sent to all the nodes. Transforms may or may not be sent to DIDO Nodes. The Transform Network represents Transform Nodes participating in the transformation of some content. Once the content transformation is complete, the changes made from the initial state of the content to the new state of the content are formulated into a Transaction and sent to all the Nodes in the DIDO Network. A Transform API is used to perform this action.

Streams

Streams are a way of subsetting or filtering DIDO Transactions using a publish/subscribe paradigm. This allows any particular Node within the Node Network not to have to listen-to or process-all the transactions presented to it. This is similar to Transforms; however, the participants in the transforms do not have to be part of the Node Network.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:2_tech_views:2-nodenet:4_msg

Last update: **2021/06/10 11:46**

2.2.3 Decentralized Finance (DeFi) Layers

return to Technical Views

The Decentralized Finance (DeFi) have defined layers 0 through 3[63]

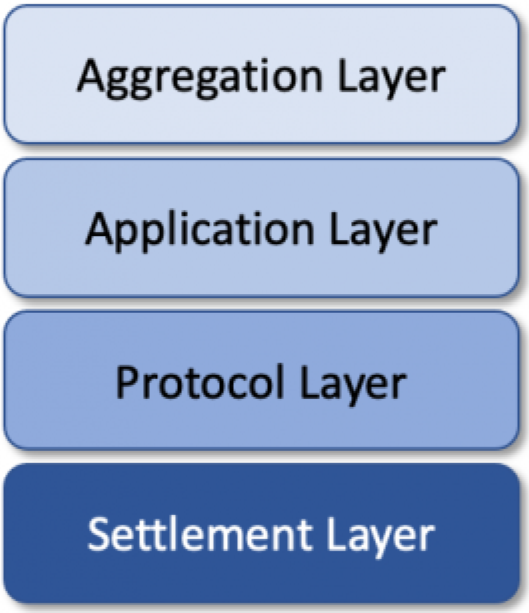


Figure 20: The Layers used in Decentralized Finance (DeFi).

Sharma defines the four layers as:

- **Settlement Layer** is also referred to as Layer 0 because it is the base layer upon which other DeFi transactions are built. It consists of a public blockchain and its native digital currency or cryptocurrency. Transactions occurring on DeFi apps are settled using this currency, which may or may not be traded in public markets. One example of the settlement layer is Ethereum and its native token ether (ETH), which is traded at crypto exchanges. The settlement layer can also have tokenized versions of assets, such as the U.S. dollar, or tokens that are digital representations of real-world assets. For example, a real estate token might represent ownership of a parcel of land.
- **Protocol Layer** are the standards and rules written to govern specific tasks or activities. In parallel with real-world institutions, this would be a set of principles and rules that all participants in a given industry have agreed to follow as a prerequisite to operating in the industry. DeFi protocols are interoperable, meaning they can be used by multiple entities at the same time to build a service or an app. The protocol layer provides liquidity to the DeFi ecosystem. One example of a DeFi protocol is Synthetix, a derivatives trading protocol on Ethereum. It is used to create synthetic versions of real-world assets.
- **Application Layer** as the name indicates, is where consumer-facing applications reside. These applications abstract underlying protocols into simple consumer-focused services. Most common applications in the cryptocurrency ecosystem, such as decentralized cryptocurrency exchanges and lending services, reside on this layer.
- **Aggregation Layer**, is the aggregation layer consists of aggregators who connect various applications from the previous layer to provide a service to investors. For example, they might

enable the seamless transfer of money between different financial instruments to maximize returns. In a physical setup, such trading actions would entail considerable paperwork and coordination. But a technology-based framework should smoothen the investing rails, allowing traders to switch between different services quickly. Lending and borrowing is an example of a service that exists on the aggregation layer. Banking services and crypto wallets are other examples.

[63] Rakesh Sharma, Investopedia, 24 March 2021, Decentralized Finance (DeFi) Definition, Accessed 24 May 2021, https://www.investopedia.com/decentralized-finance-defi-5113835

2.3 Taxonomic Views

return to Architectural Views

A taxonomy is a way of organizing things into useful and convenient classes. Each class can then be treated as an abstraction of the individual elements. For example, **Dog** is a class of **Animal**. Once an individual entity is classified as a **Dog**, generalizations can be made about what to expect from the individual entity. There can be many taxonomies that classify entities. For example, one taxonomic classification places an individual in the animal kingdom; however, another taxonomy classifies an individual according to their state of employment (working, retired, etc.). Each is valid; the only difference is perspective.

The DIDO RA employs the following four taxonomies:

- 2.3.1 Network Topology Taxonomy
- 2.3.2 Network Access Control Taxonomy
- 2.3.3 Node Taxonomy
- 2.3.4 Data Taxonomy

2.3.1 Network Topology Taxonomy

return to Taxonomic Views

Probably one of the most significant differences between DIDO architectures and other architectures is the simple, but powerful, topology of the network of nodes it uses to connect peers. DIDOs rely primarily on a distributed rather than a decentralized or centralized topology. Each network topology defines a community of nodes that act as peers, which collectively provide a solution to a problem that is then distributed. This community of nodes also employs redundant data storage, redundant computing, or both.

Most DIDO implementations rely heavily on data and computational power that exists externally and, therefore, is beyond the primary focus of the DIDO. For example, an account holder in a cryptocurrency DIDO application may require information such as currency exchange rates, the holder's nation of origin, tax IDs, or certificates of trust. In a greenfield development, all this external data would be held within the DIDO. The reality is that it is not possible to build everything from scratch. Therefore, this data might be held in other network topologies (see section 2.3.4.2 Ancillary Data).

These three kinds of network topologies are represented graphically in the following figure.

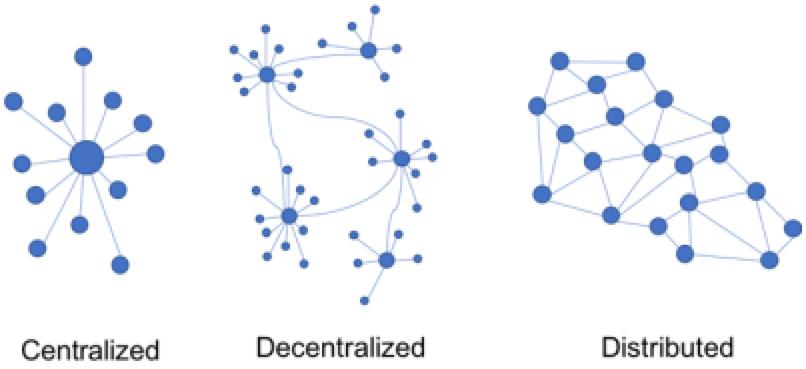


Figure 21: The Difference between Centralized, Decentralized, and Distributed Network Topologies

- 2.3.1.1 Centralized Network Topology
- 2.3.1.2 Decentralized Network Topology
- 2.3.1.3 Distributed Network Topology

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:1_topologies

Last update: **2021/05/11 05:55**

2.3.1.1 Centralized Network Topology

return to Network Topology Taxonomy

The centralized network topology (usually associated with mainframes) has a very large centralized server node that all other nodes connect to. It provides most of the services such as data storage, processing, backups, and computational power for the other nodes. It is possible to implement a “ledger” using the centralized model. In many ways, the centralized model that holds a single ledger is easier to implement and maintain since there is only one version of the data in one place and consequently, by definition, the data is the canonical data (i.e., authoritative or standard data).

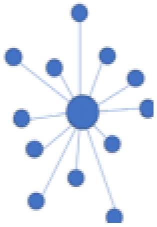
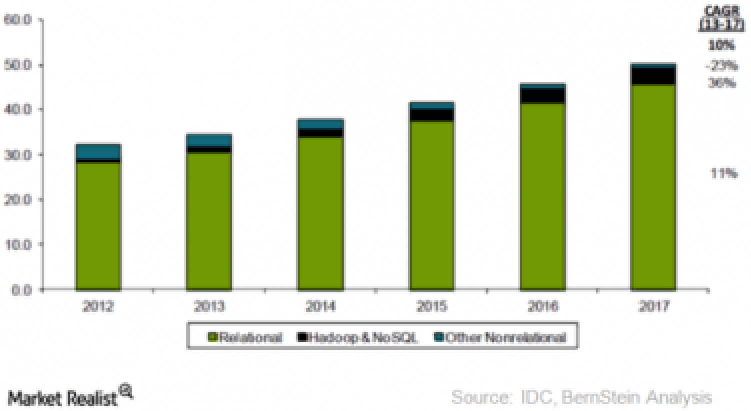


Figure 22: Centralized Network Topology

Trying to remove redundant, replicated, or duplicate data is a fundamental principle of the centralized model. The multiple “copy of data” problem was first formally addressed by E. F. Codd in his development of A Relational Model of Data for Large Shared Data Banks[64] and has become the cornerstone of Relational Database Management Systems (RDBMS), used extensively today in products such as Oracle Corporation’s Oracle, IBM’s DB2, or Computer Associates INGRES [65] . The following chart shows that the centralized database market is not shrinking, and by 2017 had grown to a net worth of $50 billion with no signs of change in this growth trend. [66] This highlights the magnitude of the effort to convert the world to DIDO technology. Though it may eventually happen, it’s going to take a long time.



Source: IDC, BernStein Analysis

Figure 23: Centralized Global Database Market ($ Billions)

[64] E. F. Codd, “A relational model of data for large shared data banks,” Communications of the ACM, vol. 13, n. 6, pp. 377-387, June 1970

[65] G2 Crowd, “Best Relational Databases Software in 2018,” 2018. [Online]. Available: https://www.g2crowd.com/categories/relational-databases [Accessed 12 June 2018].

[66]

A. Shields, “Is Oracle's Position Secure in the Database Space?,” 18 January 2016. [Online]. Available: http://marketrealist.com/2016/01/oracles-position-secure-database-space/. [Accessed 22 July 2017].

2.3.1.2 Decentralized Network Topology

return to Network Topology Taxonomy

In a decentralized network topology, there are several “central” nodes, each providing redundancy and failover capabilities for the other. Often, each of the centralized nodes are geographically distributed (i.e., North American, Europe, Southeast Asia, etc.).

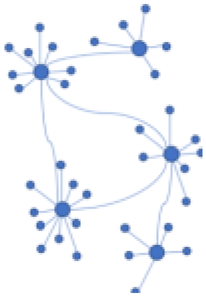


Figure 24: Decentralized Network Topology

Decentralized systems overcome some of the problems associated with a centralized system: they do not have a single point of failure, resulting in fault tolerance.[67],[68] Thus, each node can be maintained independently, which ultimately results in a more stable system. Moreover, the load on the system can be reduced by increasing the number of centralized nodes thereby distributing the work load.

Nevertheless, the scalability of the system is moderate, since the cost of expansion per node is generally steep. Granted, much of this cost has come down with the availability of Platform as a Service (PaaS) offerings from Amazon, Microsoft, and others. Since the nodes are arranged in clusters around one of the centralized servers, there is only partial fault tolerance with the result that occasionally parts of the system are rendered unavailable. The dependence on the underlying network topology means that, depending on which network links are broken, there is a chance that the data within one of the servers is obsolete.

The decentralized model is used extensively in cloud computing, specifically Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform-as-a-Service (PaaS). Although there are implementations of cloud computing which do not use the decentralized model (e.g. blockchains), the following chart from Gartner summarizes the projected growth of the “as a Service” offerings and indicates that it is on the rise. [69] It also highlights the magnitude of trying to “convert” the world to one that is completely distributed. Even were it to occur, it’s going to take a long time.

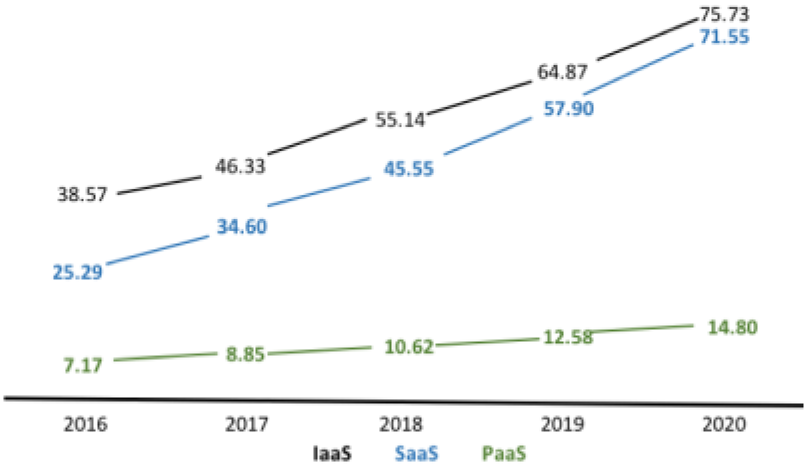


Figure 25: Decentralized Cloud Market Revenue ($ Billions)

[67] Nonetheless, the October 2016 Distributed Denial of Service (DDoS) attack on Domain Name System (DNS) servers on the US East coast and in Europe highlighted a weakness in the decentralized model.

[68] A. Shields, “Is Oracle's Position Secure in the Database Space?,” 18 January 2016. [Online]. Available: http://marketrealist.com/2016/01/oracles-position-secure-database-space/

[69] C. Coles, “Overview of Cloud Market in 2017 and Beyond,” 2016. [Online]. Available: https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/. [Accessed 24 July 2017].

2.3.1.3 Distributed Network Topology

return to Network Topology Taxonomy

In the distributed network topology, all nodes are equal peers within the system, with each acting as a redundant copy of other nodes. This provides for extremely high fault tolerance, tied to the number of nodes. The more nodes, the greater the tolerance to faults. Since each node is an equal peer, there is no single point of failure and there is no one master copy of the data. As a result, the distributed system becomes almost infinitely scalable.

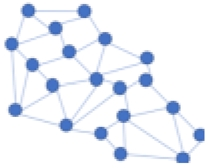


Figure 26: Distributed Network Topology

However, all nodes are not really considered “equal” in the truest sense. Some nodes are used simply to create a simple transaction, others record all the information within the blockchain, and others go on to validate transactions by mining.

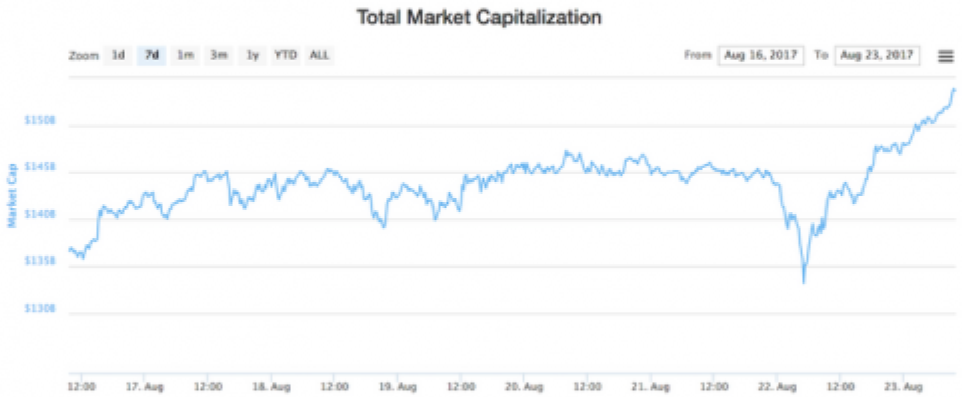


Figure 27: Market Cap of Cryptocurrencies ($ Billions)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:1_topologies:3_distributed

Last update: **2021/03/29 10:46**

2.3.1.4 Relevant Networking Standards

return to Network Topology Taxonomy

The following standards are applicable to all network views.

Technical Standards

- RFC0793 - Transmission Control Protocol
- RFC2104 - Keyed-Hashing for Message Authentication (HMAC)
- RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication
- RFC2818 - HTTP Over TLS (HTTPS)
- RFC0791 - Internet Protocol (IPv4)
- RFC2460 - Internet Protocol, Version 6 (IPv6) Specification
- RFC6749 - The OAuth 2.0 Authorization Framework
- RFC6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC1112 - Host Extensions for IP Multicasting
- RFC2315 - Cryptographic Message Syntax
- RFC3447 - PKCS #1: RSA Cryptography Specifications
- RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0
- RFC2246 - The TLS Protocol
- RFC0768 - User Datagram Protocol (UDP)

de facto Standards

- None identified

2.3.2 Network Access Control Taxonomy

return to Taxonomic Views

Network access control taxonomy classifies the types of access individuals (i.e., nodes) have from outside and from within the node network. The two main classes of access control are permissionless and permissioned.[70]

Within each of these two classifications it is possible to have public and private access. Public and private access define who is able to write data onto a network or ledger. In contrast, open (i.e., permissionless) and closed (i.e., permissioned) determine who is able to read the data. Networks are classified as[71]:

- public and open
- public and closed
- private and open
- private and closed

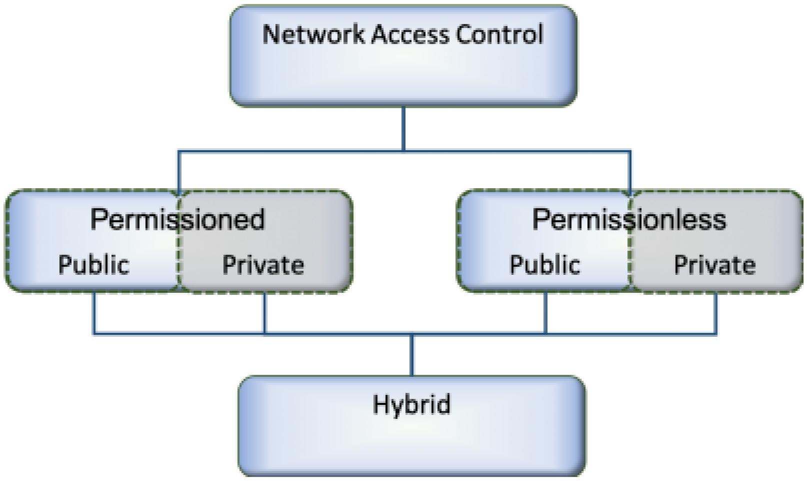


Figure 28: The Node Network Access Taxonomy

Another category of networks is a hybrid network, which makes it possible to restrict the visibility of information on the network using a combination of public, private, permissionless and permissioned networks. Therefore, hybrid networks are appealing to regulated markets because they offer the benefits of public blockchain and private blockchain together.[72]

Table 1: Taxonomy of DIDO Access Control

Permissionless Networks	Permissioned Networks
Public Networks	Private Networks
Hybrid Networks	

[70], [71] "Public Vs Private Blockchain In A Nutshell", Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

[72] "Hybrid Blockchain: Decentralized Option for Highly Regulated Markets - Few players in highly regulated markets have adopted blockchain technology. However, hybrid blockchain will change this.", Mina Down,

14 November 2018
https://blog.goodaudience.com/hybrid-blockchain-decentralize-highly-regulated-markets-900f30a37903

2.3.2.1 Permissionless Networks

return to Network Access Control

Permissionless Networks require no permission to use them. In other words, there is no barrier to entry. Anyone can run a node, run mining software/hardware, access a wallet, and write data onto and transact within the blockchain (as long as they follow the rules of the blockchain). There is no way to censor anyone, ever, on a permissionless Bitcoin blockchain.[73)]

Benefits of Permissionless Networks

Decentralized

Permissionless networks are decentralized and distributed. In other words, no one entity can close or terminate the network, modify the content, or censor parts of it. The larger the distributed and decentralized networks and or history are, the harder it is to tamper with.[74)]

Transparency

Users or nodes have complete access to the ledger, transactions, and blocks in the blockchains, which allows for complete auditing of permissionless networks[75)].

Anonymity

In permissionless networks, users or nodes of the network are anonymized. Technically, permissionless networks like Bitcoin are pseudonymous, and not truly anonymous.[76)]

Governance

As a general rule, permissionless networks rely on open source software, which is ruled by open source communities (see Talk Openly Develop Openly (TODO)). The governance of the network is by consensus. Consensus is different for many of the permissionless networksm(i.e., Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA), etc). [77)]

Tokens

Permissionless blockchains employ fat protocols that compensate network contributors with

Tokens. As the value and utility of the network increases, the value of the underlying tokens increases as well. This is the premise of cryptoeconomics and Initial Coin Offering (ICO) based fundraising. There are two predominant types of tokens today: monetary value tokens and utility tokens. Monetary value tokens are used in myriad ways as instruments for exchanging value. Utility tokens are akin to loyalty points: they have intrinsic value but no monetary value outside of that ecosystem.[78]

Scalability and Performance

For all the value blockchains bring to modern business processes, their Achilles heel often involves scalability and performance. Both Bitcoin and Ethereum blockchains suffer from poor scores in this area. For example, a recent blockchain game called Crypto kittles clogged the Ethereum network. Having said that, these are just early teething troubles, and startups are experimenting with various strategies to address this issue. Hopefully it is only a matter of time before this issue becomes a non-entity.[79]

[73] “Permissioned vs. Permissionless blockchains: Who will win and will it matter?”, Dustin D, 22 March 2018, Permissionless

[74], [75], [76], [77], [78], [79] “Nuances Between Permissionless and Permissioned Blockchains”, Anant Kadiyala, 18 February 2018, https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483

2.3.2.2 Permissioned Networks

return to Network Access Control

Permissioned Networks (Glossary Definition)

Permissioned blockchains combine the properties of both the public network and private network. Each permissioned network is unique and represents a careful balance of public and private networks to meet specific business use cases.

The available options include allowing anyone to join the permissioned network after suitable verification of their identity, and the allocation of select and designated permissions to perform only certain activities on the network. [80)]

Benefits of Permissioned Networks

Decentralization

The degree of decentralization for permissioned networks is a business decision. The extent and quality of decentralization depends upon the number of peers (i.e., nodes), the expected number of bad nodes in the network, and the type of consensus mechanism determined by the stakeholder. Permissioned blockchains usually employ an algorithm such as Byzantine Fault Tolerance, which differs from the proof of work (PoW) algorithm[81)].

Transparency

Transparency is not a driving force in permissioned networks and is often a major factor in the business decision to choose permissioned over permissionless networks. Most permissioned blockchains do not use cryptoeconomic coins incentive or tokens. The primary incentive of permissioned blockchain participants is to minimize the transparency, cost, time, and ease of sharing information[82)].

Privacy

Permissioned blockchains offer fine-grained visibility into transaction details, as well as, metadata about those transactions which, in many ways, compromises the privacy of the Network participants[83)].

Governance

There are fundamental differences between permissionless and permissioned network governance. Permissioned governance is decided and agreed upon by members of the business network. Economic incentives, code quality, code changes, and power allocation among peers are based on the business dynamics and the common purpose and goals of the permissioned members. This allows for agile and responsive networks desired by businesses[84].

Tokens

Permissioned blockchains generally do not employ a cryptoeconomic coins incentive or tokens[85].

Scalability and Performance

Permissioned blockchains use consensus mechanisms, which are computationally inexpensive (when compared to proof of work (PoW)). Therefore, they enjoy substantially better scalability and performance than their permissionless network cousins[86].

[80] “Public, Private, Permissioned Blockchains Compared”, Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

[81], [82], [83], [84], [85], [86] “Nuances Between Permissionless and Permissioned Blockchains”, Anant Kadiyala, 18 February 2018, https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483

2.3.2.3 Public Networks

return to Network Access Control

Public networks are distributed and open, allowing participation in core activities of the network from any node sponsored by anyone. Participation includes joining, leaving, reading, writing, and auditing the activities on the network. There are no authorities or discrimination of nodes.

Benefits of Public Networks

Open Read and Write[87]

> Anyone can participate by submitting transactions to the blockchain, such as Ethereum or Bitcoin; transactions can be viewed on the blockchain explorer.

Ledger Is Distributed[88]

> The database is not centralized like in a client-server approach, and all nodes in the blockchain participate in the transaction validation.

Immutable[89]

> When something is written to the blockchain, it can not be changed, in other words it is immutable.

Secure Due to Mining (protection from the Fifty-One Percent (51% Attack)[90])

> For example, with Bitcoin, obtaining a majority of network power could potentially enable massive double spending, and the ability to prevent transaction confirmations, in addition to other potentially malicious acts.

=-
=-

[87] “Public Vs Private Blockchain In A Nutshell”, Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

[88], [89], [90] “Public Vs Private Blockchain In A Nutshell”, Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

2.3.2.4 Private Networks

return to Network Access Control

A **private network** limits access to the network to individuals or nodes granted and verified to have permission to participate in joining, leaving, reading, writing, and auditing activities on the network. A participant joins a private network only through an authentic and verified invitation; validation is performed either by the network operator(s) or using a clearly defined set of protocols implemented by the network.[91)]

Benefits of Private Networks

Enterprise Permissioned[92)]:

The enterprise controls the resources and access to the blockchain, hence private and/or permissioned.

Faster Transactions[93)]:

When you distribute the nodes locally, but also have far fewer nodes that participate in the ledger, performance is faster.

Better Scalability[94)]:

Being able to add nodes and services on demand can provide a great advantage to the enterprise.

Compliance Support[95)]:

As an enterprise, you would likely have compliance requirements to adhere to; having control of your infrastructure enhances ability to satisfy this requirement more seamlessly.

Consensus More Efficient (fewer nodes)[96)]:

Enterprise or private blockchains have fewer nodes and usually a different consensus algorithm, such as BFT vs PoW.

[91)] "Public, Private, Permissioned Blockchain Compared", Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

[92], [93], [94], [95], [96]
“Public Vs Private Blockchain In A Nutshell”, Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:2_network_access_ctrll:private

Last update: **2021/03/29 10:47**

2.3.2.5 Hybrid Networks

return to Network Access Control

A **Hybrid** **Blockchain** is unique in that it is decentralized while also making it possible to restrict the visibility of information on the network with a combination of public, private, permissionless and permissioned Networks. Thus, a hybrid blockchain is appealing for regulated markets as it offers the benefits of public blockchain and private blockchain together.[97)]

The hybrid blockchain's decentralized, secure, transparent, and immutable nature provides benefits similar to those offered by permissionless and public networks: it allows for restrictions on rights to view, modify, and append/approve transactions to approved participants. In simple words, if a network member does not want their transaction data to be visible or accessible without their permission, they can earmark particular rights to view, modify, or get into consensus with different members.[98)]

Benefits of Hybrid Networks

Private Transactions

Transaction are private but verifiable using the ledger's immutable data objects (i.e., leverage its public state). In its public state, each transaction gets approved by a massive network and is essentially secure and trustworthy. Hence, there is no need for a central governing body or an exhaustive chain of intermediaries to supervise things. So, any change done to a transaction will undergo a “kindred” approval process, making it next to impossible for a single actor to meddle with the transaction or entries[99)].

Equality

Everyone in the network has equal rights to view, modify, and append their consent to a transaction. In addition, the identity of transacting parties is never disclosed to all the visible network participants. [100)].

Non-Repudiation

Anonymity is simply not acceptable to financial institutions and regulated industries with their strict Know Your Customer (KYC) standards. [101)].

Confidentiality

Unrestricted visibility of the public state of the network exposes all the data to a colossal network breach, which is counter to data confidentiality obligations, as well as their business concerns.[102)].

[97] “Hybrid Blockchain: Decentralized Option for Highly Regulated Markets - Few players in highly regulated markets have adopted blockchain technology. However, hybrid blockchain will change this.”, Mina Down, 14 November 2018 https://blog.goodaudience.com/hybrid-blockchain-decentralize-highly-regulated-markets-900f30a37903

[98], [100], [101], [102] “If you Thought Blockchain was Amazing, Wait till You Read about Hybrid Blockchain”, Atul Khekade, 20 January 2018, https://www.entrepreneur.com/article/307794

[99] “If you Thought Blockchain was Amazing, Wait till You Read about Hybrid Blockchain”, Atul Khekade, 20 January 2018, https://www.entrepreneur.com/article/307794. This article uses the term “agnate approval” rather than “kindred approval”; however, agnate limits a kindred relationship to males only. Thus, we prefer the term “kindred” over “agnate”

2.3.3 Node Taxonomy

return to Taxonomic Views

A node[103] is any participant in a DIDO Node Network. However, there are different types of nodes within the node network. The types are classified according to the kind of activities (i.e. roles) they need or want to perform within the node network.

All nodes must provide their own hardware to participate. The hardware requirements can range from fairly simple handheld devices to larger servers capable of storing vast amounts of data and processing many transactions. However, servers are not essential for the operation of the node network since all participants operate as peers within a peer-to-peer (P2P) network of equals (i.e., it is not a client-server model). The correct value of data (i.e. “Truth”) is achieved through consensus and results in each node having a local copy of the “true” values. Thus, there is no need to go to a central server to get the true, accurate, or current values of the data.

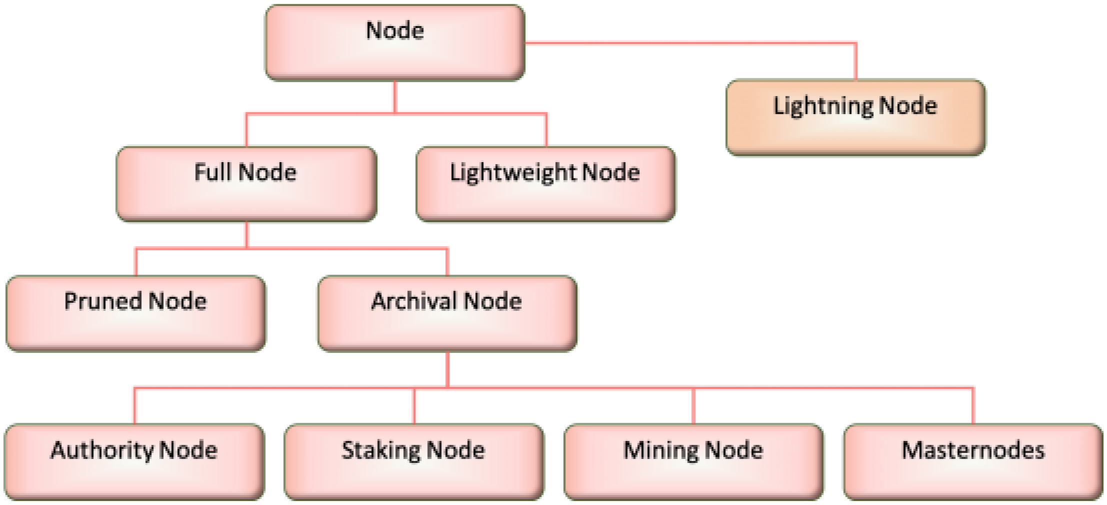


Figure 29: DIDO Node Taxonomy: Node Types

The classifications shown in Figure 1 are generalizations; each domain may define more or fewer types of nodes with each assigned different roles than those described here. For example, Bitcoin really defines only two kinds of nodes: full nodes and lightweight nodes. Full nodes have the entire copy of the ledger and can create transactions on their own. Lightweight nodes must work with a full node in order to synchronize the current “true” value of the data. Bitcoin also has a mining node, which is a full node used to validate that a block of transactions is “true.” Sometimes DIDO nodes can be referred to as clients[104] and are classified by their level of engagement with the DIDO; however, any domain can define or modify the definitions for node types.

<table>
<tr><td colspan="3">Full Node</td><td>Lightweight</td><td colspan="2">Lightening</td></tr>
<tr><td>Pruned</td><td colspan="4">Archival</td><td>Permanode</td></tr>
<tr><td></td><td>Authority</td><td>Staking</td><td>Mining</td><td>Master</td><td></td></tr>
</table>

[103] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

[104] **Clients** are nodes able to parse and verify blockchain ledger, transactions, and smart contracts. Clients have access to APIs with which to create transactions and mine blocks. https://ethereum.stackexchange.com/questions/269/what-exactly-is-an-ethereum-client-and-what-clients-are-there

2.3.3.1 Full Node

return to Node Taxonomy

Full nodes keep a full copy of the blockchain transactions[105]. There can be any number of full nodes within the node network, all acting as redundant data sources. Some of the activities of a full node are maintaining consensus between other nodes, verification of transactions, and storing the ledger.

In many ways the full nodes' functionality is analogous to those of servers in decentralized networks. However, there is no centralized “truth” or final judge. Instead, the “truth” is determined by consensus among the full nodes. However, this consensus based methodology is not without its pitfalls. When more than 51% of the full nodes cannot reach consensus (i.e., agree with a transaction or a proposition), the proposed change is skipped. This can lead to a Hard Fork in the ledger and the opposing groups diverge, creating two or more chains [106]. Sometimes the 51% problem can be part of an orchestrated effort, referred to as a 51% attack [107]. The more nodes in the node network, the harder it is to successfully launch a 51% attack.

A well-known example of this kind of ledger divergence, leading to a hard fork, was the Bitcoin Cash Fork.[108]

Full node contains:

- 2.3.3.1.1 Pruned Node
- 2.3.3.1.2 Archival Node

[105] Osita Chibuike, 21 May 2018, Legobox, https://dev.to/legobox/how-to-setup-an-ethereum-node-41a7

[106] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

[107] “51% Attack”, Jake Frankenfield, 6 May, 2019, https://www.investopedia.com/terms/1/51-attack.asp

[108] “Bitcoin Cash’s Scheduled Hard Fork Tripped Up By Software Bug”, Christine Kim, 15 May 2019, https://www.coindesk.com/bitcoin-cash-scheduled-hard-fork-tripped-up-by-software-bug

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:3_node_tax:full

Last update: **2021/03/29 10:48**

2.3.3.1.1 Pruned Node

return to Full Node

A **pruned node** is a full node for all intents and purposes with one major difference: in order to save space on the node, the pruned node begins downloading blocks from the beginning of the ledger and when a preset threshold on space is exceeded, the oldest transactions are deleted, retaining only the headers and placement within the blockchain.

For example, a size limit threshold of 550MB allows storage of the latest blocks that fit within the 550MB threshold. However, to retain the integrity of the ledger, all the transactions must be processed and checked for validity in the order of their creation. At the end of the load, the state of the data is correct and only the history is of the transactions is lost.[14]

Note: Pruned nodes are considered full nodes and thus can also verify transactions and be involved in consensus.

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- None at this time

[14] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

2.3.3.1.2 Archival Node

return to Full Node

In more traditional DIDOs, an **archival full node** is primarily responsible for storing the entire ledger and for providing consensus to the entire node network by validating blocks and potentially receiving a reward [110].

As the size of the ledgers grow, there will be fewer archival nodes and more pruned nodes. IOTA refers to these archival nodes as “permanodes” and believes that business services will evolve around either charging for the storage or charging to retrieve the data that is stored or perhaps both. In some cases, such as in the Industrial Internet of Things (IIoT), the nodes that publish data might be rewarded when or if the the data is retrieved.

> **Note:** The difference between pruned and archival nodes is that pruned nodes only keep the latest validated data and require far less storage on the local node.

Archival nodes are divided into subtypes, three that can add blocks to a blockchain and one that cannot:

- Can add blocks: authority node, staking node, and mining node
- Cannot add blocks: masternode

[110] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/. Article covers the various kinds of “rewards” a node can receive. Descriptions of the Archival Node subtypes also covers rewards.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:3_node_tax:full:archival

Last update: **2021/03/29 10:48**

2.3.3.1.2.1 Authority Node

return to Archival Node

Traditionally, the participation in DIDO node networks to perform tasks is permissionless, requiring no outside authority from anyone or anything. Having permissionless access to a node network is one of the original guiding principles that is integral to the decentralized nature of DIDOs[111].

> **Note:** Unfortunately, there are drawbacks to this approach. Solutions involving a level of centralization for granting permission can provide benefits like increased speed because there is no need for the costly, time consuming consensus algorithms such as Delegated Proof of Stake (DPoS), Delegated Byzantine Fault Tolerant (dBFT), Proof of Authority (PoA) and others. However, centralized permissioned systems are vulnerable to malicious attacks such as denial of service, thereby undermining many of the "democratizing" aspects of the DIDO.

Networks making use of PoA networks define a fixed number of **authority nodes**. The number of nodes and associated PoA designation is voted on by the PoA community or defined by the development team. Authority nodes are similar to full nodes and can create and validate blocks. All other nodes in the node network run as lightweight nodes, depending on broadcasted data to participate in the blockchain.[112]

Iota refers to authority nodes as **permanodes** (definition) because they keep all transaction data even after snapshots are made (concept introduced by Iota [113])

[111] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[112] "Blockchain Nodes: An In-Depth Guide", https://nodes.com/

[113] "Full node vs permanode", Helmar, 5 January 2018, https://iota.stackexchange.com/questions/782/full-node-vs-permanode/783

2.3.3.1.2.2 Staking Node

return to Archival Node

A **staking node** is usually a lightweight node (wallet). It allows for the aggregation of individual stakeholders stakes in Proof of Stake (PoS) a node network. The collection of tokens (i.e., coins) has a higher weight [114] and improves the chances of receiving more rewards for validating a block of transactions. [115]

Characteristics of Staking Nodes[116]

- Ease of setup (basically a lightweight node (wallet))
- No initial startup costs
- No payout delay
- No penalty for being offline
- No minimum balance
- No guarantee of returns
- Usually smaller returns than a masternode

[114] See Weight of Network

[115] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

[116] “Masternode vs Staking”, Solaris Support Center, 26 June 2019, https://solaris.helpsite.com/articles/24861-masternode-vs-staking

2.3.3.1.2.3 Mining Node

return to Archival Node

Mining nodes [117] [118] are full nodes or lightweight nodes within the node network that attempt to prove they have worked and completed the required challenge before anyone else to create a block of transactions (see proof of work (PoW)). To complete the task, mining nodes perform either as full archival nodes, or receive data from other full nodes on the node network to obtain the current state of the blockchain and parameters required to describe the next block.

Mining nodes employ hardware components (i.e., CPUs, GPUs or ASICs) to solve a cryptographic problem. The first mining node to complete the task broadcasts the results to the node network for verification by full nodes. Once consensus is achieved on the determination that the solution to the problem is correct and the solution is the first to be posted, the mining node is granted the right to add a block to the existing blockchain.

As a reward for performing the work, mining nodes are given a preset number of tokens (i.e., coins), as well as the transaction fees associated with the block of transactions. The preset reward is often referred to as a *coinbase* or a *coinbase transaction*. The reward (i.e., coinbase) is usually the first transaction in the block and free. [119],[120]

[117] Not all DIDOs use “mining” as originally defined in Bitcoin as Proof of Work (PoW). This means other DIDO implementations (i.e., IOTA) may not require a full blockchain Ledger to verify a transactions validity.

[118] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 24 May 2009. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[119] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

[120] Osita Chibuike, 21 May 2018, Legobox, https://dev.to/legobox/how-to-setup-an-ethereum-node-41a7

2.3.3.1.2.4 Masternode

return to Archival Node

A **masternode**[121] is combination of a staking node using a Proof of Stake (PoS) node network, which relies on the weight of the stake[122], and a server.

The requirements for a successful staking node are a server, a stable internet connection, a minimum number of coins used for staking, and time to mine the server. Unlike the staking node, the masternode does not wait for randomly assigned blocks of transactions to validate but is instead constantly engaged: obtaining rewards and constantly paying out a certain number of tokens (i.e., coins) (thus the minimum stake).[123]

Characteristics of Masternodes[124]

- Higher rewards than a simple staking node
- Ability to participate in votes on proposals
- Hosting does not have to be local
- Higher cost and resource requirements than staking node
- More complexity
- Requires an initial stake

[121] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

[122] See Weight of Network

[123] “What's the difference between staking and masternode?”, darkangel11, 03 February 2018, https://bitcointalk.org/index.php?topic=2874856.0

[124] “Masternode vs Staking”, Solaris Support Center, 26 June 2019, https://solaris.helpsite.com/articles/24861-masternode-vs-staking

2.3.3.2 Lightweight Node (Wallet)

return to Node Taxonomy

Lightweight nodes keep a shallow copy of blockchain transactions[125] and are used in day-to-day crypto operations. Within the cryptocurrency world, lightweight nodes are also referred to as simple payment verification (SPV) nodes or wallet nodes. Lightweight nodes communicate with the ledger through full nodes.[126]

Standards

Technical Standards

- ISO/IEC 9899:2018 Programming languages -- C
- ISO/IEC 14882:2017 Programming languages -- C++
- ECMA: Standard ECMA-262 - ECMAScript® 2018 Language Specification (Javascript)
- ECMA: Standard ECMA-334 - C# Language Specification
- ECMA: Standard ECMA-335 - Common Language Infrastructure (CLI)
- ECMA: Technical Report TR/84 - Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File
- ECMA: Technical Report TR/89 - Common Language Infrastructure (CLI) - Common Generics
- RFC5424 - The Syslog Protocol (SYSLOG)
- W3C: RDF 1.1 Terse RDF Triple Language (Turtle)
- W3C: OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (second Edition)
- W3C: RDF 1.1 Concepts and Abstract Syntax (RDF)
- W3C: SPARQL 1.1 Overview (SPARQL)
- W3C: Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification
- W3C: HTML5 (HTML5)
- W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition)
- W3C: XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C: XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- W3C: XSL Transformations (XSLT) Version 3.0
- W3C: Document Object Model (DOM) Level 3 Core Specification
- W3C: XML Path Language (XPath) 3.1
- OMG: Data Distribution Service (DDS)
- OMG: DDS Interoperability Wire Protocol (DDSI-RTPS)
- OMG: ISO/IEC C++ 2003 Language DDS PSM (DDS-PSM-Cxx)
- OMG: Java 5 Language PSM for DDS (DDS-Java)
- OMG: OPC-UA/DDS Gateway (DDS-OPCUA)
- OMG: RPC Over DDS (DDS-RPC)
- OMG: DDS Security (DDS-SECURITY)
- OMG: Web-Enabled DDS (DDS-WEB)

- OMG: DDS Consolidated XML Syntax (DDS-XML)
- OMG: DDS For Extremely Resource Constrained Environments (DDS-XRCE)
- OMG: Extensible and Dynamic Topic Types for DDS (DDS-XTypes)

de facto Standards

- Apache: Log4j
- Apache: Log4cxx
- Apache: log4php
- Apache: log4net
- Apache: log4jscala
- Bitcoin: Developer's Guidance
- Ethereum: cpp Project
- Ethereum: Ethereumh Project
- Ethereum: Ethereumjs-lib Project
- Ethereum: Ethereum_j Project
- Ethereum: Go-ethereum Project
- Ethereum: Parity Project
- Ethereum: Pyethapp Project
- Ethereum: Ruby-ethereum Project
- EIP 20: ERC-20 Token Standard
- Ethereum: Ethereum Virtual Machine (EVM)
- Ethereum: Solidity Language Specification
- Linux Foundation: Hyperledger
- Oracle: The Java® Language Specification SE 8 Edition
- Oracle: The Java® Virtual Machine Specification JVM
- Oracle: Java logger API
- Google: Go (software language)
- InterPlanetary File System (IPFS)

Tools

- None at this time

[125] Osita Chibuike, 21 May 2018, Legobox, https://dev.to/legobox/how-to-setup-an-ethereum-node-41a7

[126] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

2.3.3.3 Lightning Node

return to Node Taxonomy

A **lightning node** is a participant in a lightning network[127] using multi-signature (multisig)[128] on a payment channel[129]. Lightning nodes create a small community (2 to n) of blockchain nodes. The community can trade without having to use the expensive ledger transaction and has been proposed as a way to provide scalability to Bitcoin.

[127] “Blockchain Nodes: An In-Depth Guide”, https://nodes.com/

[128] “How to Use Multisig to Keep Your Coins Ultra-Safe”, Kai Sedgewick, 2 March 2019, https://news.bitcoin.com/how-to-use-multisig-to-keep-your-coins-ultra-safe/

[129] “Bitcoin’s Lightning Network Payment Channel Explained !!”, Sudhir Khatwani, 11 March 2019, https://themoneymongers.com/payment-channels/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:3_node_tax:lightening

Last update: **2021/03/29 10:49**

2.3.3.4 Permanode

return to Node Taxonomy

Permanode is an IOTA-unique term for an authority node, a variant of an archival node that retains transaction data even after a snapshot is taken. It is not shown in the node taxonomy figure in Section 2.3.3 Node Taxonomy.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:3_node_tax:perma

Last update: **2021/03/29 10:49**

2.3.4 Data Taxonomy

return to Taxonomic Views

The third word in the acronym DIDO is *Data* and every node therefore, manages and controls data. The data within the node is classified as either: ledger data, ancillary data or external data.

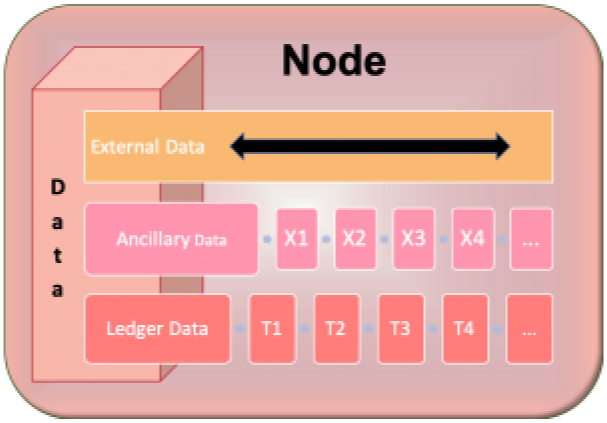


Figure 30: Types of Node Data

- Ledger Data
- Ancillary Data
- External Data

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- None at this time

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

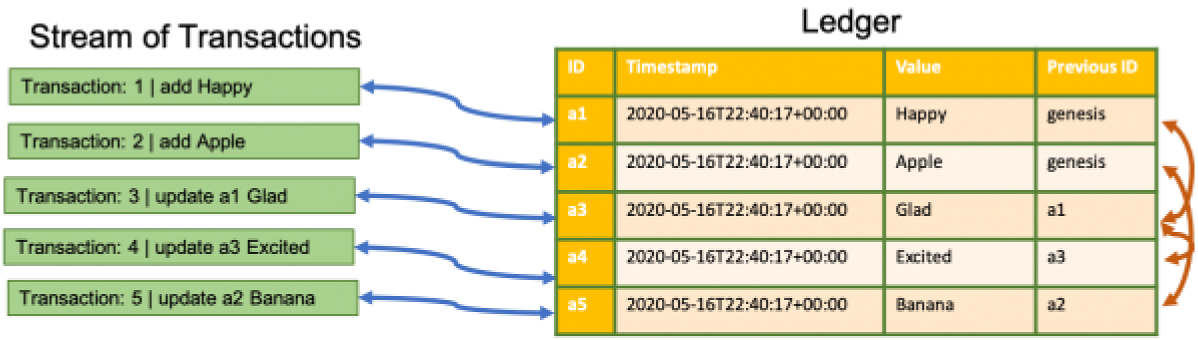
Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:4_data_tax

Last update: **2021/05/11 05:55**

2.3.4.1 Ledger Data

return to Data Taxonomy

Ledger data is comprised of records containing immutable data. Even though the data captured within the ledger is immutable, this does not mean the ledger itself is immutable. The ledger can be updated by adding newer versions of existing data that then point back to the original data. Updates are made by applying transactions to the ledger in a prescribed order.



Stream of Transactions

Transaction: 1 | add Happy

Transaction: 2 | add Apple

Transaction: 3 | update a1 Glad

Transaction: 4 | update a3 Excited

Transaction: 5 | update a2 Banana

Ledger

ID	Timestamp	Value	Previous ID
a1	2020-05-16T22:40:17+00:00	Happy	genesis
a2	2020-05-16T22:40:17+00:00	Apple	genesis
a3	2020-05-16T22:40:17+00:00	Glad	a1
a4	2020-05-16T22:40:17+00:00	Excited	a3
a5	2020-05-16T22:40:17+00:00	Banana	a2

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- None at this time

2.3.4.2 Ancillary Data

return to Data Taxonomy

Ancillary data is adjunct information used in the formulation of transactions. It differs from transaction data because it is not used to define the operation described within a transaction. Probably the best way to explain ancillary data is using examples.

Examples of Ancillary Data

Supporting Data

Data needs to be exchanged between two individuals that describe the exchange of one kind of currency with another (e.g., US dollars to EU euros or Bitcoins). The transaction can simply be something like this:

```
TRANSFER 500 USD to 1 Bitcoin in MyAccount
```

The transaction seems legitimate and can probably be verified easily enough, or can it? In order to verify the transaction, the exchange rate used also needs to be provided. It might also require the timestamp of when the exchange rate was calculated and even where the exchange rate was obtained from. This "extra" information used to verify the transaction is called ancillary data. It ultimately needs to be coded into the transaction. A classic example of why ancillary data needs to be captured is to avoid salami slicing.

Business Process Management

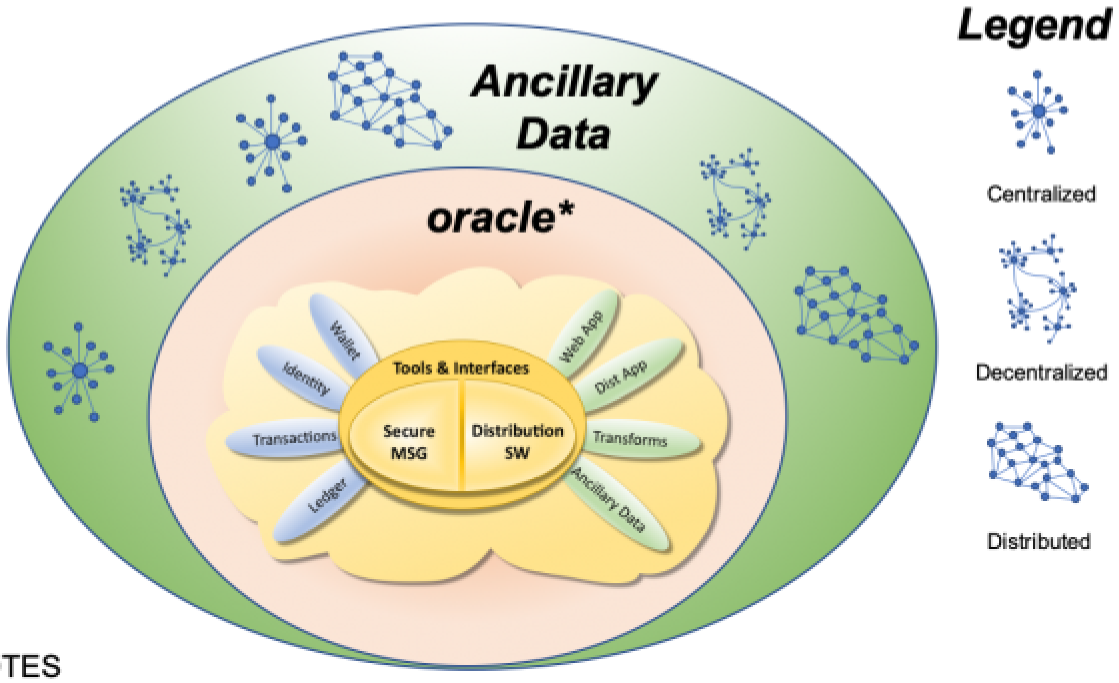
Data needs to be exchanged between two different business entities. Each business entity has its own business process describing the steps needed on its side in order to approve a transaction. The data required to approve the transaction is transformational in nature rather than transactional.

```
TRANSFER 200 AMZN Shares from MyOrg to YourOrg
```

At a transactional level, all that is needed is verification that MyOrg has the shares, and YourOrg exists in order to prevent "double spend" fraud, i.e., spending the same money more than once. However, many organization have business processes in place that require signatures from various people and that these signatures must be in a prescribed sequence (i.e., Director signs, VP signs, Comptroller signs). To verify the transaction, the signatory data needs to be provided and confirmed that it is valid. This can occur outside the transaction and be transformational in nature. Transformational data can be undone, deleted, or modified until the business process commits to the the transaction.

DIDO Implementation of Ancillary Data

Some DIDO implementations may provide direct support of ancillary data within the DIDO or they may provide access to external ancillary data through oracles. However, if ancillary data is not also implemented within the DIDO, but implemented externally through and accessed through the use of oracles, many of the benefits of the distributed architecture are lost because access to the ancillary data can become a bottleneck.



NOTES
* An oracle, in the context of blockchains and smart contracts, is an agent that finds and verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts. https://blockchainhub.net/blockchain-oracles/

Standards

Technical Standards

- None at this time

de facto Standards

- None at this time

Tools

- None at this time

2.3.4.3 External Data

return to Data Taxonomy

External data is data that is neither ledger data nor ancillary data but is required within the blockchain or smart contracts. The external data is accessed using an oracle. There are numerous formats for external data sources: data files, databases, rich site summary (RSS) feeds, RESTful interfaces, etc.

Standards

Technical Standards

- OMG: Data Distribution Service (DDS)
- RFC8259 - The JavaScript Object Notation (JSON) Data Interchange Format
- W3C: HTML5 (HTML5)
- Linux Foundation: Open Messaging
- Linux Foundation: Open Middleware Agnostic Messaging API (OpenMAMA)

de facto Standards

- None at this time

Tools

- None at this time

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.2_views:3_taxonomic:4_data_tax:3_external

Last update: **2021/03/29 10:49**

3 Governance

return to Reference Architecture

For practical reasons, the acronym Distributed Immutable Data Object (DIDO) represents a set of distributed computing technologies that focus on distributed data (i.e., blockchains, distributed ledgers, distributed file systems or distributed data). A major problem confronting the adoption of DIDO technologies is that it requires shifting away from corporate models of governance to an open community model of development, especially for industry or public ecosystems (e.g., financial, supply chains, public records) and domains (e.g., interest swaps, produce supply chain, cryptocurrencies, carbon credits, air pollution, traffic conditions). In the corporate model of governance, a single entity is responsible for the costs and the entire lifecycle of a product or project. Governance is accomplished through a formal chain of command, usually with a single individual responsible for the success or failure of the product or project.

However, in distributed computing, not all the resources are owned, paid for, or controlled by a single entity. In fact, the more entities involved in the distributed computing solution, the better. These differences in governance models make the adoption of distributed computing difficult by corporate entities since they have to rely on a larger, more inclusive community, which may include competitors, to measure the success of the solution. Ultimately, the success of the project or product comes down to controlling and minimizing risks. Being part of a larger, more diverse community increases some kinds of risks but may decrease others.

For example, the risks associated with specifying, architecting, designing, implementing, testing, maintaining, and sunsetting code are shared over the entire community, thus practically lowering the risks to any individual. The risks associated with setting requirements, priorities, and so on may be increased since these are now set externally to any single entity.

Currently, with the rise of open source solutions versus proprietary solutions, many efforts have moved from the centralized governing corporate model towards the decentralized, distributed community governing model with a great deal of success and broad adoption. Some examples of open source solutions are Operating System (OS), DataBase Management System (DBMS), application servers, web servers, file repositories, bug tracking tools, virtualization tools, and]]dido:public:ra:xapend.glossary:d:dlt]]. Most corporations would now rarely choose to build any of these products on their own, but have readily adapted to the adoption of these open source community products and solutions.

This section defines and recommends community of interest (CoI) governance structures needed for successful, robust, inclusive, and broadly adopted solutions. Some CoIs require a formal legal entity recognized by a government authority such as a state government. Other CoIs can operate within the confines of another legal CoI. For example, World Wide Web Consortium (W3C) and Object Management Group (OMG) are legal entities. Special interest groups (SIGs) or working groups can operate within the W3C or OMG.

- 3.1 DIDO Communities
- 3.2 Legal Documents
- 3.3 Guides

Manage an Open Source Program

There have been many books written on this subject:[130]

- RFC2026 - The Internet Standards Process
- TODO: Using open source code
- TODO: Participating in open source communities
- TODO: Recruiting open source developers
- TODO: Starting an open source project
- TODO: Improve your open source development impact
- TODO: Shutting down an open source project
- TODO: Building leadership in an open source community
- TODO: Setting an Open Source Strategy

[130] TODO Open Source Reading List, https://todogroup.org/guides/open-source-reading-list/

3.1 DIDO Communities

Return to the Governance page

As introduced in Section 2.1, a DIDO community is a community of interest (CoI) responsible for the architecture, design, implementation, maintenance, and eventual sunset (yes, all good things come to an end) of DIDO networks and its DIDO nodes. DIDO communities should have a well-documented, repeatable, traceable, transparent, and unbiased process that is managed and run by a governing board (or hierarchy of governing boards). Often the DIDO community is divided into two organizations: one responsible for the software, usually as an open source software (OSS) project, and the other responsible for managing the fungible data (e.g., currency or commodity) managed by the software. Most DIDO communities choose or select DIDO software designed, implemented, and maintained by another party. For example, Ethereum, Hyperledger, and IOTA all provide their software for others to use as OSS.

Some DIDO communities may support multiple kinds of immutable data objects within a single DIDO network, or they can support multiple networks that each have different kinds of immutable data object types. For example, one DIDO community's OSS might support cryptocurrencies on one DIDO network and support immutable data objects representing supply chain commodities on another.

Unless a DIDO network's focus is very narrow, there is generally a need to have an exchange that allows different immutable data objects to be exchanged (e.g., US dollars to Bitcoins). The exchange might support the exchange of immutable data objects that are all contained within one DIDO network, or it might support the exchange of information managed by one DIDO community but on different DIDO networks. In this case, the exchange might manage converting customer loyalty reward points to a currency (i.e., euros or cryptocurrency) or supply chain commodities to cryptocurrencies and back. Other exchanges might exchange immutable data objects managed by one DIDO community and DIDO network with a cryptocurrency managed by another DIDO community; for example, the exchange of Bitcoins to Ethereum or Iotas.

- 3.1.1 Stakeholder Communities
- 3.1.2 Software Communities

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.3_gov:1_communities

Last update: **2021/03/29 10:50**

3.1.1 Stakeholder Communities

Return to DIDO Communities Page

Given that DIDOs are centered around stakeholders, a stakeholder community is key to creating a thriving community of stakeholders. Unless the target of a DIDO is a private, permissioned network, there will be stakeholders that exist outside of any corporation and/or a corporation's customers or clients. In many cases, the stakeholder community will also include competitors or other participants (i.e., supply chain players) that want to integrate services around a common concept or idea. This common concept or idea is generally a distributed object.

In other words, the stakeholder community is generally defined as people, groups, organizations, or businesses that have interest or concern in the distributed object. Stakeholders affect or are affected by the community's actions, objectives and policies.

A stakeholder community can be informal or formal. Informal communities are best described as a confederation of the willing and generally have one or more core participants who make decisions for the group. When the stakeholder community is large, broad and with sometimes competing stakeholders, these communities should be formal, well organized organizations (usually non-profit).

Note: Refer to Section 2.1 Stakeholder Views for how the various stakeholder communities are organized, as well as the definitions and descriptions of ecosphere, ecosystem, and domain. Also, it should be understood that each of these terms refer to a kind of community of interest (CoI). Thus, saying “ecosphere” should be read as “ecosphere CoI”. Likewise for “domain” and “ecosystem”.

Steps for Establishing an Ecosphere

Steps for establishing and building a formal ecosphere CoI[131]

1. Define a Mission Statement
2. Select a Leadership Team as Governing Board
3. Secure Funding
4. Create Legal Documents:
    a. Charter
    b. Bylaws
    c. Policy and Procedures (P&P)
5. Legally Incorporate
6. If Non-Profit, officially register as Non-Profit (In US, 501 ©(3))
7. Identify Partners (Founding Members)
8. Establish Milestones, Deliverables, and Schedules
9. Create Community Guides consistent with Charter, Bylaws, and Policies and Procedures
10. Recruit more members
11. Establish liaisons with other CoIs

[131]

How To Start A Nonprofit Organization, Snowball Fundraising, Accessed 21 May 2020, https://snowballfundraising.com/starting-a-nonprofit-checklist/

3.1.2 Software Communities

Return to DIDO Communities Page

The cornerstone of a DIDO community is the software, which acts as the engine of the distributed network of peers. Software is responsible for maintaining the ledgers on each node: securely and reliably sending transactions to each of the nodes and mitigating any conflicts that may arise while processing these transactions (e.g., the double spend problem). As a general rule, the software is open source software (OSS), governed by OSS rules for its maintenance and the development of new features.

In some communities, the software extends beyond the core software required to maintain the node within a network of nodes. For example, the Ethereum Foundation includes both the software to maintain the network of nodes and smart contracts to monitor and oversee transactions. However, smart contracts themselves may or may not be built, maintained, and supported by the Ethereum Foundation's software team. They may be created, maintained, and supported by external third parties.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.3_gov:1_communities:2_software

Last update: **2021/03/29 10:50**

3.2 Legal Documents

return to Governance

Given the intentional distributed, decentralized nature of DIDOs, along with the potential for them to run on hardware and/or use software not owned or developed by a single entity, it is important to employ a community of interest (CoI) approach to develop, manage, and govern DIDO solutions. Such an approach allows for the establishment and specification of requirements for a formal organization that balance the various goals of its participants in pursuit of a single shared mission and, at the same time, protects the joint intellectual property (IP) resulting from their collaboration.

- **Note:** Legal Documents serve as the Regulatory Aspect within the Governing Model. See Governing Roles and the Governing Model for more information.

When an organization is a joint venture between different legal entities, it must operate in accordance with legal documents that serve to manage the expectations of, govern the interactions between, and elaborate the collective and individual deliverables expected from the legal entities participating in that organization. In the context of DIDO, *ecosphere* CoI is the term used to refer to such a joint venture. Table 3 lists the legal documents required to formally launch an ecosphere CoI. These documents support the process described in stakeholder communities. Depending on the country of incorporation, all three may be required for an ecosphere COI to be formally incorporated. In general terms (more details provided in Sections 3.2.1, 3.2.2, and 3.2.3), the charter defines how the ecosphere CoI interacts with the outside world; the bylaws define how it operates internally (e.g., fundamental governing rules, internal organization); and the policies & procedures (P&P), a document required for incorporation in the United States, establishes a set of principles to achieve the goals and vision of its charter and the specific methods employed to express these policies in action.

As already discussed in stakeholder views, other types of CoIs or special interest goups (SIGs) can be formed under the umbrella of a parent ecosphere: *ecosystems* and *domains*.[132] As shown in Table 3, they must each be chartered according to the governing rules established in the bylaws and abide by the P&P of their parent ecosphere CoI. Ecosystem P&Ps may include specific extensions to the parent ecosphere's CoI P&P. In like manner, domain P&Ps may include specific extensions to the their parent ecosystem CoI P&P. All extensions to the parent P&P must be **additive**; they may not replace or override anything in the P&P of their parent ecosphere CoI.

Table 3: Documents required to Create and Govern a DIDO CoI

DIDO CoI	Charter	Bylaws	Policies and Procedures
Ecosphere	Yes(§)	Yes(†)	Yes(‡)
Ecosystem	Subcharter of Ecosphere	covered by Ecosphere	covered by Ecosphere + extensions
Domain	Subcharter of Ecosystem	covered by Ecosphere	covered by Ecosphere + extensions from Ecosystem _ local

- *((§) Initially, a legal statement created by the founders of the organization that lays out the goals, missions and officers for the organization*
- *(†) a legal document reviewed by lawyers from all the participating parties*
- *(‡) Some Policies and Procedures may be mandated by law (i.e., discrimination, ADA, Safety, etc.*

while others may be added by local governing boards and should be drafted/reviewed by lawyers of all participating parties

NOTE: There are no standards for these documents. The initial versions of these documents, particularly the Bylaws, should be drafted by attorneys. In the case of the P&P, the initial version should be drafted with input from attorneys to address legally mandated items.

- 3.2.1 Charter
- 3.2.2 Bylaws
- 3.2.3 Policies and Procedures (P&P)

[132] Described in more detail, along with illustrations, in Ecosystem View and Domain View.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.3_gov:1_legaldocs

Last update: **2021/06/09 00:05**

3.2.1 Charter

return to Legal Documents

A charter is a legal document providing basic information about a community of interest (CoI): its location, purpose[133], profit status (usually non-profit), governing board (e.g., board of directors) composition, and stakeholder (or ownership) structure. Sometimes the charter is referred to as the articles of incorporation or a certificate of incorporation. In the case of a for-profit corporation, the articles of incorporation must include the number, classes, and par values of authorized shares. In the United States, most states require the name and address of the company's registered agent as well. Source: Charters: Creating the Organization

The CoI charter is important because when it is filed and approved by a legal entity, such as a secretary of state, it "gives birth" to a new CoI as a formal corporation. A CoI charter is not the same as the CoI bylaws, which build off the charter and add things like fundamental governing rules, board meeting schedules, conditions of membership, etc.

Note:

1. Charter, as described in this section, only applies to ecosphere CoIs that file for incorporation.
2. Ecosystem CoIs and domain CoIs are also created using a charter, but these charters are not legal or legally binding documents. Instead they are "subcharters" of, and must be approved by, their parent ecosphere using a process defined in the parent ecosphere's P&P.
3. The Charter serves as part of the Regulatory Aspect within the Governing Model. See Governing Roles and the Governing Model for more information.

Essential Elements of a Charter

- **Name**
- **Address**
- **Statement of Purpose**
- **Profit Status (Usually non-Profit)**
- **Registered Agent**
- **Agent's Address**
- **Number of Shares Authorized**
- **The Classes and par value of the shares**
- **Roles** and **Responsibilities**
- **Directors**

Standards

- The charter ID made specifically for the CoI (i.e., ecosphere).

Tools

- Smartsheet Project Charter Template, Smartsheet, 21 May 2020, https://www.smartsheet.com/blog/project-charter-templates-and-guidelines-every-business-need
- UpBoard, Project Charter Online Tools & Templates – Best Practices, accessed 21 May 2020, https://upboard.io/project-charter-online-software-tools-templates/

References

- The Essential Guide to Creating an Effective Team Charter
- Investing Answers, Corporate Charter, Accessed 21 March 2020, https://investinganswers.com/dictionary/c/corporate-charter
- Investopedia, Understanding a Corporate Charter, Accessed 30 May 2020 https://www.investopedia.com/terms/c/corporatecharter.asp

[133] in the context of a DIDO CoI, think of this as another word for “Goal”, “Mission”, or “Vision”

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.3_gov:1_legaldocs:1_charter

Last update: **2021/03/29 10:50**

3.2.2 Bylaws

return to Legal Documents

The community of interest (CoI) bylaws (or articles of organization) are the primary legal documents providing governance for the group. The bylaws are originally created during the initial steps of establishing the CoI (i.e., ecosphere). [15]

Bylaws are supplemental to the rules already defined by the granting government entity's (e.g., states, counties, or cities in the US) code and rules for how the CoI must run. Bylaws specify the rules and regulations governing actions and decisions made by the board. The bylaws should help prevent or resolve conflicts and disagreements, and protect the CoI from potential problems by clearly outlining rules concerning authority levels, rights, and expectations.

> **Note**:
>
> 1. If the board of directors fails to follow the adopted CoI bylaws, it can be held liable for breaching their duty to the CoI.
> 2. The Bylaws serves as part of the Regulatory Aspect within the Governing Model. See Governing Roles and the Governing Model for more information.

Common Provisions in Bylaws

1. Name and Purpose
2. Parliamentary Authority for Election, roles, and terms of officers and board members
3. Membership Issues (categories, responsibilities)
4. Meeting Guidelines (Frequency and Quorum)
5. Board Structure and Size
6. Compensation and indemnification of board members
7. Role of Chief Executive
8. Conflict of Interest Policy
9. Amendment of Bylaws Policy
10. Dissolution of the organization

Standards

- The bylaws are made specifically for the CoI (i.e., ecosphere).

de facto Standards

- Robert, Henry M.; et al. (2011). Robert's Rules of Order Newly Revised (11th ed.). Philadelphia, PA:

Da Capo Press. ISBN 978-0-306-82021-2 (hardcover).
- Robert III, Henry M. (2011). “The Official Robert's Rules of Order Web Site (Home)”. The Official Robert's Rules of Order Web Site. The Robert's Rules Association.
- The Official Robert's Rules Of Order Web Site (robertsrules.com) Site maintained by the Robert's Rules Association

Tools

- Section 7. Writing Bylaws, Community Tool Box, Accessed 21 May 2020, https://ctb.ku.edu/en/table-of-contents/structure/organizational-structure/write-bylaws/main
- Form of Bylaws - California Nonprofit Public Benefit Corporation, Public Coucil Org, Accessed 21 May 2020, http://www.publiccounsel.org/tools/publications/files/Annotated_Bylaws.pdf
- Sample Bylaws, U.S. Chamber of Commerce, Accessed 21 May 2020, https://www.uschamber.com/your-chamber-commerce-guide-starting-and-growing-chamber-commerce/sample-bylaws

References

- Bylaws for an Unincorporated Association, Drew Nelson, September 26, 2017, Accessed 21 May 2020, https://bizfluent.com/list-6981569-bylaws-unincorporated-association.html
- What is the difference between Charter and Bylaws?, William Adkins, bizfluet, 26 September 2017, Accessed 21 May 2020, https://bizfluent.com/facts-5894520-difference-between-charter-bylaws-.html
- The Difference Between Bylaws and Policy, Victoria Duff, 26 September 2017, Accessed 21 May 2020, https://bizfluent.com/facts-5921586-difference-between-bylaws-policy.html
- Bylaws, Policies and Procedures: What's the Difference?, Karen Blewett, presented to Board Leadership Southeast Alberta, 4 March 2017, Accessed 30 May 2020 https://boardleadershipsouth.com/files/march_4_2017/Bylaws,%20Policy%20and%20Procedures%202017.pdf

[15] Nonprofit Bylaws Made Easy: Tips and Best Practices, donorbox.org, Acccessed 21 May 2020, https://donorbox.org/nonprofit-blog/nonprofit-bylaws-made-easy/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.3_gov:1_legaldocs:2_bylaws

Last update: **2021/03/29 10:50**

3.2.3 Policies and Procedures (P&P)

return to Legal Documents

Policies and Procedures (P&P)

One of the legal documents required for the formal incorporation of an ecosphere community of interest (CoI) is the Policies and Procedures (P&P). The P&P establishes a set of principles and policies that serve these principles, in order to achieve the CoI's long-term goals (i.e, the purpose as stated in its charter). The P&P defines specific methods/procedures employed to express these policies in action: detailed rules and guidelines to control the activities of the organization. In the US, some policies are concerned with human resources (HR) issues (e.g., Health Insurance Portability and Accountability Act (HIPAA), or data protection[16] and security), a direct response to the General Data Protection Regulation (GDPR), Data Protection Act 2018, or the California Consumer Privacy Act (CCPA).

The intent of the P&P is to influence and help formulate all the major decisions and actions of the organization. All activities within the organization are to take place within the boundaries set by the P&P.[17] Procedures are specific methods outlining the steps required to fulfill the policies on a day-to-day basis within the organization. Together, policies and procedures help the governing body of an organization meet its mission and goals.[18]

One of the larger policies that is often required by the governing body granting incorporation are the human resources policies, especially those concerning health and human safety.

- **Note** The Policies and Procedures (P&P) serve as part of the Regulatory Aspect within the Governing Model. See Governing Roles and the Governing Model for more information.

Standards

- The charter ID made specifically for the CoI (i.e., ecosphere).

Laws

- Occupational Safety and Health Act of 1970, Public Law 91-596, 84 STAT. 1590, 91st Congress, S.2193, December 29, 1970, as amended through January 1, 2004. https://www.osha.gov/laws-regs/oshact/completeoshact
- Regulation (EU) 2016/679 OF THE European Parliament and of the Council of 27 April 2016 https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

Tools

- Rocket Lawyer, Health and Safety Policy Document Maker, Accessed 21 may 2020,

https://www.rocketlawyer.com/gb/en/documents/health-and-safety-policy
- Rocket Lawyer, Data Protection and Data Security Document Maker, Accessed 21 May 2020, https://www.rocketlawyer.com/gb/en/documents/data-protection-and-data-security-policy

References

- Health and Safety, Rocket Lawyer, accessed 21 May 2020. https://www.rocketlawyer.com/gb/en/quick-guides/health-and-safety
- Overview of the Data protection and data security policy, Rocket Lawyer, accessed 21 May 2020. https://www.rocketlawyer.com/gb/en/documents/data-protection-and-data-security-policy

[16]

NOTE: From https://iclg.com/practice-areas/data-protection-laws-and-regulations/usa,

- *There is no single principal **Data Protection** legislation in the United States. Rather, a jumble of hundreds of laws enacted on both the federal and state levels serve to protect the personal data of U.S. residents. At the federal level, the **Federal Trade Commission Act (15 U.S. Code § 41 et seq.)** broadly empowers the U.S. Federal Trade Commission (FTC) to bring enforcement actions to protect consumers against unfair or deceptive practices and to enforce federal privacy and data protection regulations. The FTC has taken the position that "deceptive practices" include a company's failure to comply with its published privacy promises and its failure to provide adequate security of personal information, in addition to its use of deceptive advertising or marketing methods. As described more fully below, other federal statutes primarily address specific sectors, such as financial services or health care.In parallel to the federal regime, state-level statutes protect a wide range of privacy rights of individual residents. The protections afforded by state statutes often differ considerably from one state to another, and cover areas as diverse as protecting library records to keeping homeowners free from drone surveillance.*
- *Although there is no general federal legislation impacting data protection, there are a number of federal data protection laws that are sector-specific, or focus on particular types of data. By way of example,*
  o Driver's Privacy Protection Act of 1994 (DPPA)
  o Children's Online Privacy Protection Act (COPPA)
  o Video Privacy Protection Act (VPPA)
  o Cable Subscriber Protection

[17]

Difference Between Policies and Procedures, Key Differences, accessed 21 May 2020,https://keydifferences.com/difference-between-policies-and-procedures.html

[18]

Policies and Procedures, Business Dictionary, http://www.businessdictionary.com/definition/policies-and-procedures.html

3.3 Guides

return to Governance

In addition to the legal documents governing the activities of a community of interest (CoI), many CoIs create a simple, easily understood guide to help its members maneuver through and understand all the governing statements contained in the statutes, charter, bylaws, parliamentary authority, special rules, custom practices, and standing rules associated with that particular CoI. For example, the OMG publishes a "Hitchhiker's Guide to the OMG Process". Even though it is not a “normative” document (has no legal standing whatsoever, cannot be cited as a defense for not following process), it is much easier to read and understand than the OMG Policies and Procedures. It provides additional details and insights that, even though they do not belong in a P&P, are essential to know. Technical standards bodies such as the IETF publish a similar guide, which is even more informal and flippant. In any case, the goal of such a guide is to make it easier for the members of a CoI to follow their CoI's process rules.

4 Requirements

return to Reference Architecture

The term requirement first appeared in software engineering in the 1960s.[138]

The Business Analysis Body of Knowledge® version 2 from IIBA (BABOK),[139] defines a requirement as having one or more of the following characteristics:

1. *A condition or capability needed by a stakeholder to solve a problem or achieve an objective.*
2. *A condition or capability that must be met or possessed by a solution or solution component to satisfy a contract, standard, specification, or other formally imposed documents.*
3. *A documented representation of a condition or capability as in (1) or (2).*[140]

This definition is based on IEEE Standardized vocabulary.[141]

- 4.1 About Requirements
- 4.2 Functional Requirements
- 4.3 Non-Functional Requirements
- 4.4 Assessing Requirements

[138] Boehm, Barry (2006). A view of 20th and 21st century software engineering. ICSE '06 Proceedings of the 28th international conference on Software engineering. University of Southern California, University Park Campus, Los Angeles, CA: Association for Computing Machinery, ACM New York, NY, USA. pp. 12–29. ISBN 1-59593-375-1. Retrieved January 2, 2013. http://dl.acm.org/citation.cfm?id=1134288

[139] 1.3 Key Concepts - IIBA | International Institute of Business Analysis“. www.iiba.org. Retrieved 2016-09-25. http://www.iiba.org/babok-guide/babok-guide-v2/babok-guide-online/chapter-one-introduction/1-3-key-concepts.aspx

[140] Wikipedia Requirement

[141] IEEE/ISO/IEC 24765-2010 - ISO/IEC/IEEE International Standard - Systems and software engineering – Vocabulary , 2010-02-02, published 15 December 2015, IEEE, [https://standards.ieee.org/standard/24765-2010.html]]

4.1 About Requirements

Return to Requirements

Although Requirements at first glance appear to be simple, they quickly become very complex as the number of requirements increase, and consequently they need to be managed and governed. Add this complexity to the complexity associated with any distribute system, and it can quickly seem daunting and overwhelming. Like any other large problem that confronts us, it is too hard to handle it as one big problem. In engineering, a common approach is to not address the one big problem but to refine the problem into a series of little problems which can be solved. Consequently, that is why the bigger solution requires governance.

The following chapters help refine this big problem into smaller ones.

- 4.1.1 Governance Requirements Model
- 4.1.2 Cognitive Requirements Model
- 4.1.3 Governing Roles - Combined Requirements Model
- 4.1.4 Example of a Using the Combined Requirements Model
- 4.1.5 The Current State of DIDO Requirements
- 4.1.6 One Degree of Freedom Rule
- 4.1.7 Specifying Requirements

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:00_aboutreq

Last update: **2021/03/10 22:55**

4.1.1 Governance Requirements Model

Return to About Requirements

In addition to this definition, requirements can be applied to various aspects of a project from a governance perspective. There are three different aspects of governance: Regulation, Execution and Compliance (See Appendix J: Governance Model. [142]

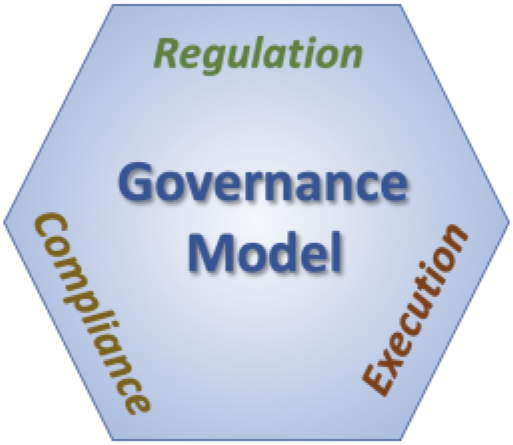


Figure 31: General Governance Model

- **Regulation** covers the specification of the requirements of the product or service. This can be either formal government regulation such as U.S. ADA compliance[143][144], or it can be project specific such as the contract or the technical specification of a contract or can be as detailed as the steps defined for a particular test plan.
- **Execution** covers the lifecycle of a product or service being governed (i.e., design, building, maintenance, etc). This covers the functional and non-functional requirements of the product or service and can be in terms of static or dynamic compliance points
- **Compliance** covers the oversight of the product or service being governed. This covers the not only the product or service itself but also the process of Regulation and Execution. For example, is the contract or technical specification well written and maintained throughout the lifecycle of the product or service.

Understanding requirements is an important part of specifying, building, using and maintaining any product. And when the System is Distributed on multiple machines, supporting multiple stakeholders the need for the proper governance is essential. The requirements must cover all three aspects of governance. For example, there can be a requirement that is expressed in the regulatory aspect which requires reporting of information. The Compliance aspect will have a corresponding requirement that validate and verifies the reporting of the information. These are then also reflected in the Execution Aspect that has to have processes and systems designed to collect and report the proper data.

[142] Stavros, Robert W. and Albrant, Jeremiah; Engineering Governance, SPAWAR, October 9, 2007,

[143] ADA Website Accessibility Under Title II of the Americans with Disability Act (ADA) - ADA Best Practices

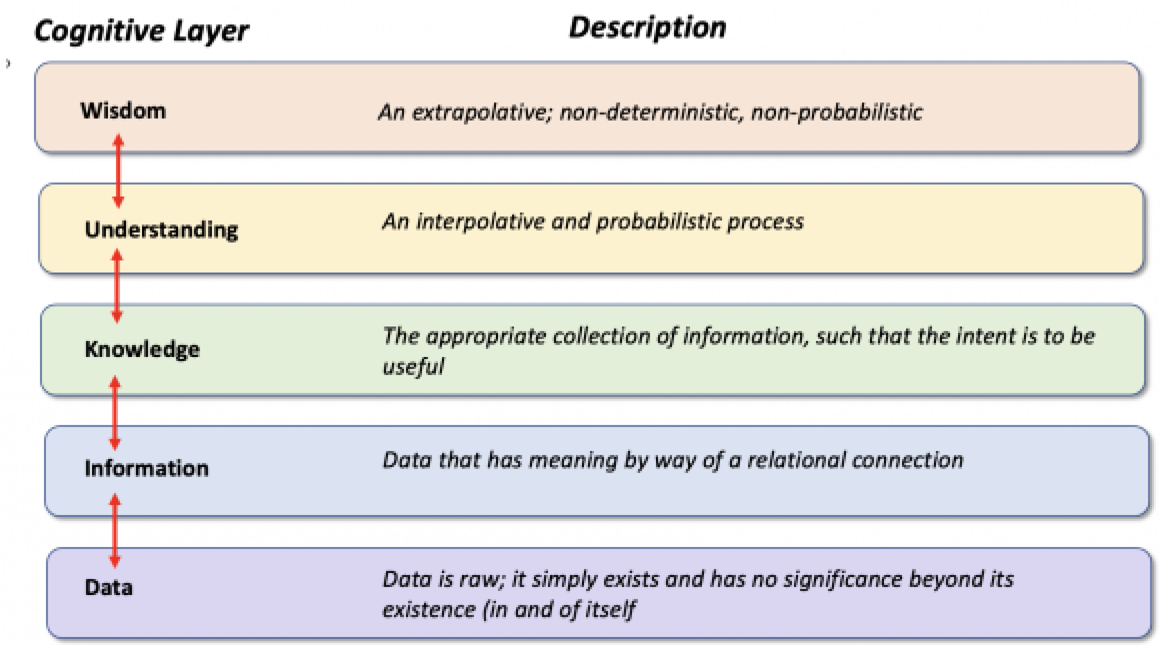
Tool Kit for State and Local Governments, Website Accessibility Under Title II of the ADA, Chapter 5, https://www.ada.gov/pcatoolkit/chap5toolkit.htm

[144] Accessibility of State and Local Government Websites to People with Disabilities, U.S. Department of Justice, Civil Rights Division, Disability Rights Section https://www.ada.gov/websites2.htm

4.1.2 Cognitive Requirements Model

Return to About Requirements

In addition to the definition of a requirement and the specification of the requirements at each of the Governance Aspects, there is a need to further refine the requirements according to the level of specificity of the requirement. There are five different layers to specificity which are based upon the layers of the Cognitive Model. (See Appendix I: Cognitive Model).



Cognitive Layer	Description
Wisdom	An extrapolative; non-deterministic, non-probabilistic
Understanding	An interpolative and probabilistic process
Knowledge	The appropriate collection of information, such that the intent is to be useful
Information	Data that has meaning by way of a relational connection
Data	Data is raw; it simply exists and has no significance beyond its existence (in and of itself

Figure 32: Cognitive Model

- **Wisdom** - An extrapolative and nondeterministic, non-probabilistic concept which is built upon **Understanding** concepts.
- **Understanding** - An interpolative and probabilistic concept that reflects **Wisdom** concepts
- **Knowledge** - A concept relating appropriate collection of **Information** concepts together, such that it's intent is useful
- **Information** - A concept aggregating data concepts together, in other words, **Data** that has been given meaning by way of relational connection
- **Data** - Data that is a raw concept; it simply exists and has no significance beyond its existence (in and of itself). It is the basis upon which **Information** concepts are built

The flow between the cognitive layers is bidirectional and can have many-to-many relationships. For example, any particular **Wisdom** concept can be associated with any number of **Understanding** concepts. The inverse is also true; **Understanding** concepts can also be used to help support any number of **Wisdom** Concepts.

4.1.3 Governing Roles - Combined Requirements Model

Return to About Requirements

In order to be effective, it is best to combine the Governance and the cognitive models together. The results look something like Figure 33. Each cell in the overlaid models represents a single Role or area of consistent governance providing some context for the requirements. For example, at the **Data** x **Regulation** cell, there is specific data that is required to be collected according to the regulations. There is a regulation that requires a bank to collect taxpayer IDs for each account. During the **Execution** aspect (i.e., the Bank's Policies and Procedures(P&P)) the taxpayer ID is collected, the specific bank actually collects and records the taxpayer ID. During the **Compliance** aspect, there is a requirement to verify that each Bank actually has a taxpayer id with each account. This consistency in governance can be repeated for each row (i.e., Wisdom, Understanding, Knowledge, Information and Data) and for each column (i.e., Regulation, Execution and Compliance).

If any of the Roles (i.e., cells have no requirements, the governance is incongruent and can lead to a potential flaw or hole in the governance which is vulnerable to exploitation.

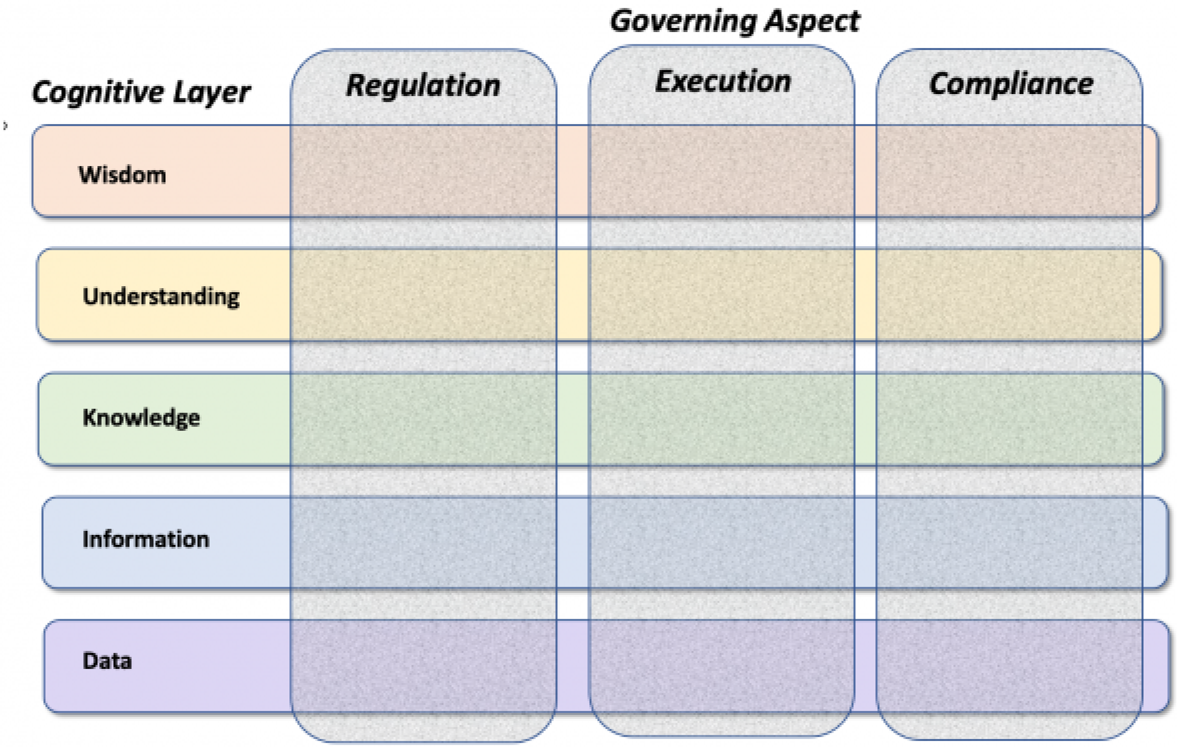


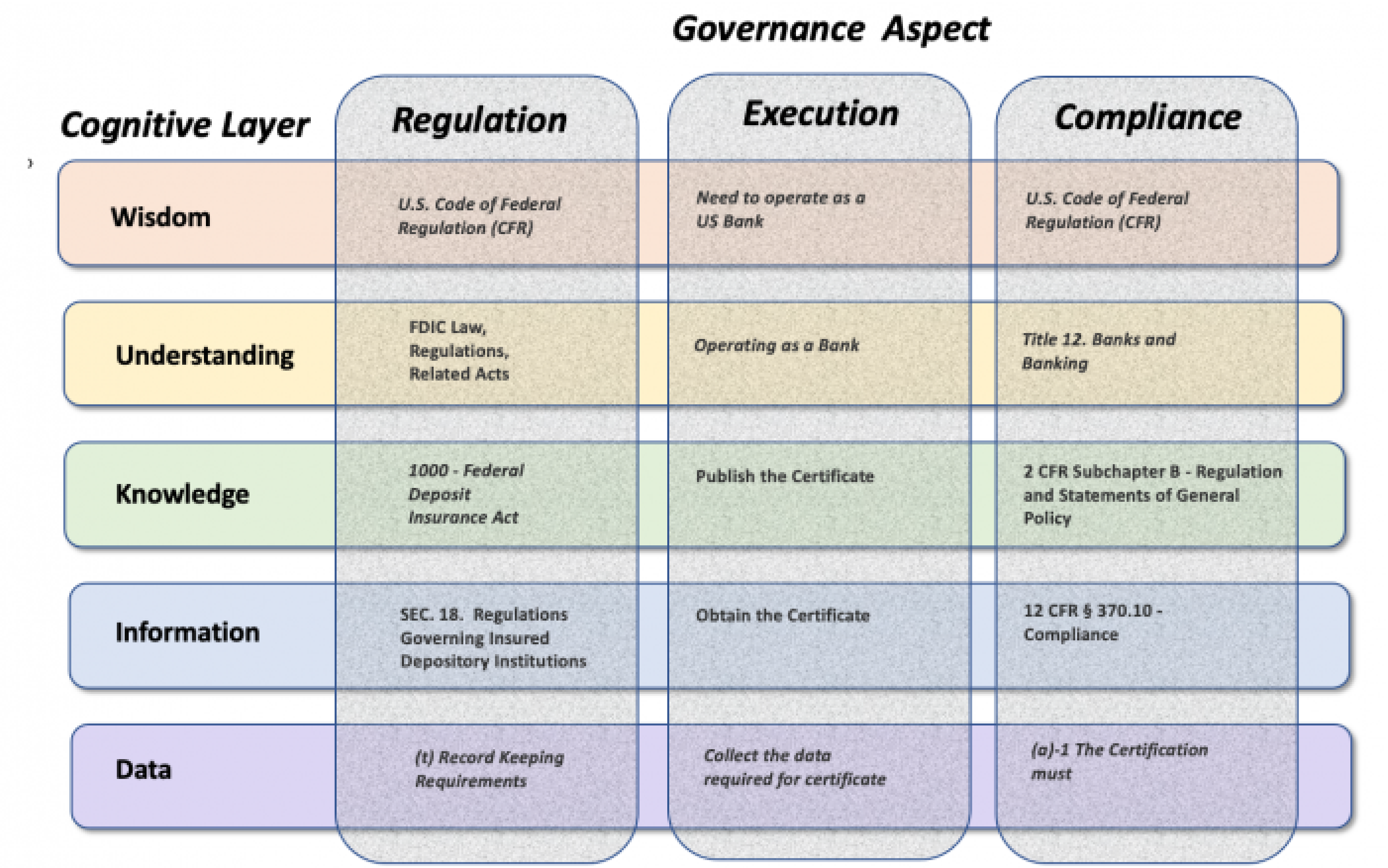
Figure 33: Combined Governing and Cognitive Models - Each Cell represents a Governing Role

4.1.4 Example of a Using the Combined Requirements Model

Return to About Requirements

The following is a Normative[145] example of using the Combined Requirements Model to review requirements for any Federal Deposit Insurance Corporation (FDIC) regulation. The actual data may be invalid or based on some convenient assumptions but is included as an example of how the combined model can be used.]

In the example, the regulation starts with **Wisdom**. Within Regulation, the **Wisdom** is the U.S. Code of Federal Regulations. Although we can make light of Federal Regulations as being **Wisdom** try and think about it this way: *“when working within the confines of a government, it is not wise to ignore or marginalize regulation”*. When working within the banking industry, **Understanding** is knowing that there are U.S. Federal Regulations that pertain to banking in general. **Knowledge** is to know the specific laws that pertain to the Area-of-Interest (AoI), in this case banking and Federal Insurance covering banking(i.e, FDIC). There is knowledge that is provided within the FDIC legislation the that specifies which laws must be followed by an institution wanting to be insured (i.e., *Section 18 of the FDIC regulations*). Finally, at the **Data** layer, there are specific rules about what needs to be reported and how it is to be reported.



	Governance Aspect		
Cognitive Layer	Regulation	Execution	Compliance
Wisdom	U.S. Code of Federal Regulation (CFR)	Need to operate as a US Bank	U.S. Code of Federal Regulation (CFR)
Understanding	FDIC Law, Regulations, Related Acts	Operating as a Bank	Title 12. Banks and Banking
Knowledge	1000 - Federal Deposit Insurance Act	Publish the Certificate	2 CFR Subchapter B - Regulation and Statements of General Policy
Information	SEC. 18. Regulations Governing Insured Depository Institutions	Obtain the Certificate	12 CFR § 370.10 - Compliance
Data	(t) Record Keeping Requirements	Collect the data required for certificate	(a)-1 The Certification must

Figure 34: Normative Example of the Combined Governing and Cognitive Models

Figure 34 roughly lays out a “straw man” of what the overall combined governance could be like. Within each governing Aspect and within each Cognitive layer there are specific requirements that covers that Role (i.e., cell).

[145] **Normative** - are statements based on opinions about what should happen. They are subjective rather than objective because they involve value judgment about what is right and what is wrong.

4.1.5 The Current State of DIDO Requirements

Return to About Requirements

The current state of DIDO governance is basically “Execution Centric”. This basically means that the governance is being developed by those that are actually creating the DIDO platforms (i.e., products). Although this approach may seem efficient as a way to overcome bureaucratic obstacles, it ultimately leads to failures in the end results. In Figure 2, the **Execution** is represented as a pyramid. This represents the bottom up approach used to developed most DIDO platforms today and is a reflection of the Agile Model Agile methodology popular on non-Mission Critical Systems.

As cryptocurrencies attempt to become financial instruments and represent actual national currencies, they are essentially positioning themselves as Mission Critical for the nations, their economies and their citizens. Therefore, to be successful, they must be governed properly which implies that each Role (i.e., cell) within the combined Governing and Cognitive models (See Figure 2) need to have an Actor (i.e. be completed).

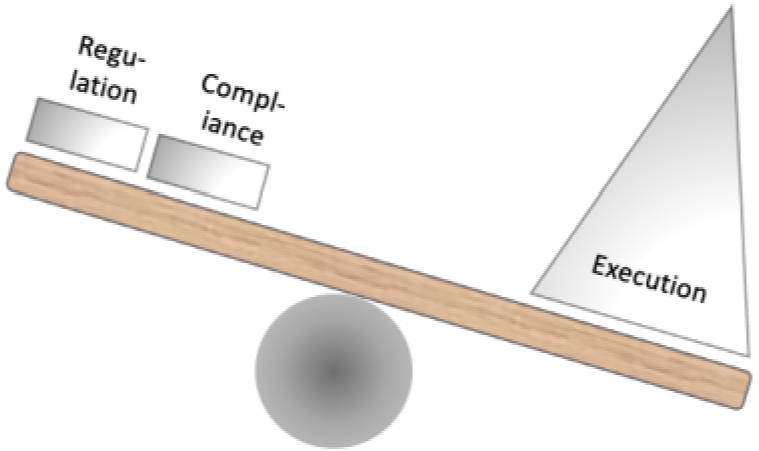


Figure 2: The current state of DIDO Governance

This is not unusual when a new disruptive product or service comes along, it is to be expected. However, in order to mature to the next level and gain wider acceptance, the product or service needs to mature its governance. The Combined Governing and Cognitive Model described and depicted in Section 4.1.3 offers a good framework for a checklist of what is done and what needs to be done.

The relationship between the Roles (i.e., cells) in a many-to-many relationship that is not only bidirectional in up and down the cognitive layers but also between the governance aspects.

4.1.6 One Degree of Freedom Rule

Return to About Requirements

Probably one of the most important rules is to not skip roles. Each and every role is important and all too often, in the name of expedience, attempts are made to short circuit the model and skip roles. For example, trying to specify in “regulatory wisdom” how to inspect products built during execution. This does not however mean that the roles are completely isolated. Within each of the Governance Aspects, the Cognitive Model’s hierarchy still applies. This can be summarized as the “one degree of freedom” rule.

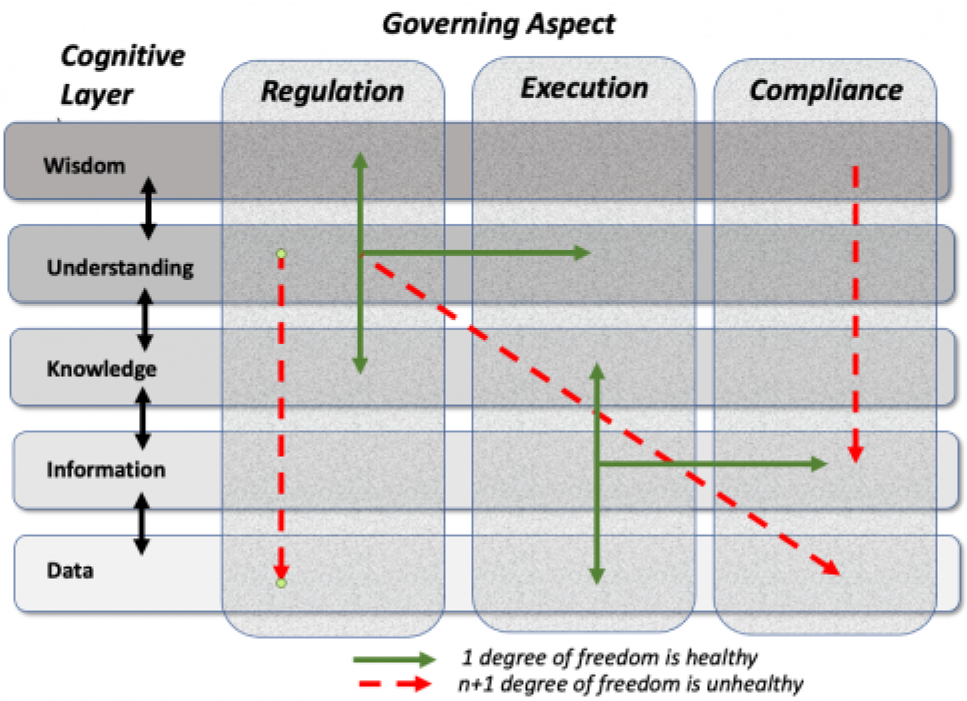


Figure 3: The One Degree of Freedom Rule

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:00_aboutreq:06_onedegree

Last update: **2021/03/07 16:18**

4.1.7 Specifying Requirements

Return to About Requirements

Requirements are captured in many ways. In the government realm, this is usually done through codification into laws, regulations and contracts including Performance and Conformance Specification. In the corporate realm, it often comes in the form of Charters, Bylaws and Policies and Procedures (P&P) and contracts with other entities (see above).

Regardless of where the requirements are captured or by what organizations, they can in general be considered governing statements. The following are some guidance on how to write healthy governing statements and consequently also requirements.

A Governance Statement based on an Engineering Governance Model developed at US Navy SPAWAR[146] is defined as atomic, succinct, absolute and definitive in nature. It contains specific instructions which can be validated through observation, measurement or testing.

- **Atomic** - A Governance Statement only addresses a single topic. Indicators of non-atomic guidance are use of highly complex sentences, multiple sentences or conjunctions such as and, or, etc.
- **Succinct** - A Governance Statement are short and to the point. The definition of terms or caveats that explain when a statement is applicable are not acceptable as part of the Governance Statement. Indicators of non-succinct statements are the use of words or expressions such as: consider, when possible, if, etc.
- **Absolute** - A Governance Statement is evaluatable with one or more non-subjective questions. Indicators of non-absolute statements are those which are subject to the interpretation of the evaluator. For example, “All menus must be user-friendly“. No one produces menu's that they feel are user hostile.
- **Definitive** - A Governance Statement is precisely worded and explicit in nature. Their words, terms and expressions need to be defined and not subject to interpretation. Indicators of non-definitive guidance are words that are not intuitively obvious to an outside reader. Some words that are examples of non-explicit words are: object, service and function.

Another issue or controversy with specifying requirements is how the statements use imperatives, words like:

- Shall (Requirement)
- Must (Requirement)
- Will (Requirement)
- Should (Requirement)

A major sticking point is the use of the word **Shall**[147] which basically claims:

- lawyers regularly misuse it to mean something other than “has a duty to.” It has become so corrupted by misuse that it has no firm meaning.
- it breeds litigation. There are 76 pages in “Words and Phrases” (a legal reference) that summarize hundreds of cases interpreting “shall.”
- nobody uses “shall” in common speech. It’s one more example of unnecessary lawyer talk. Nobody

> says, “You shall finish the project in a week.”

However, the counterargument is that ***Must should not be use because no one has defined how must is different from shall. Also, shall has held up in court, must has not.*** [148]. So, whether you should use **Shall** or **Must** is a matter for the Community of Interest (CoI) to determine, and it should be consistent.

There is an excellent reference in How to Write and Exceptionally Clear Requirements Document[149].

[146] Stavros, Robert W. and Albrant, Jeremiah; Engineering Governance, SPAWAR, October 9, 2007,

[147] PlainLanguage.gov, Shall and Must, Accessed 5 March 2021, https://www.plainlanguage.gov/guidelines/conversational/shall-and-must/

[148] Wheatcraft, Lou, Requirement Experts, 9 October 2021, Accessed 5 March 2021, https://reqexperts.com/2012/10/09/using-the-correct-terms-shall-will-should/

[149] QRA, How to Write and Exceptionally Clear Requirements Document, Accessed 5 March 2021, https://qracorp.com/write-clear-requirements-document/#elementor-toc__heading-anchor-1

4.2 Functional Requirements

Return to Requirements

Functional Requirements define the basic system behavior. Essentially, they are requirements stating what the system does or must not do, and can be thought of in terms of how the system responds to inputs. Functional requirements usually define if/then behaviors and include calculations, data input, and business processes.

Functional Requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements. Functional Requirements can be used during all phases of a project Lifecycle independent of the development model (i.e., Waterfall Model or Agile Model). In the Waterfall method, these requirements are generally specified early on in the process. In the Agile method, they can be applied throughout each Sprint or applied during specific Sprints.

The DIDO RA views functional requirements from the following perspectives:

- 4.2.1 Platforms
- 4.2.2 Access Control

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:1_func

Last update: **2021/06/09 16:10**

4.2.1 Platforms

Return to Functional Requirements

A **Platform** is an overloaded term and depends on the context it is used in. Sometimes, Platform refers to just the hardware (i.e., x86, 68000, CISC, RISC, ARM, etc.), other times it can refer to the Operating system (i.e., Windows, Linux, MacOS, Android, iOS), sometimes it can refer to the run-time environment provided by the programming languages used (i.e., C, C++, C#, Java or .NET), while othertimes it can refer to the networking used to connect computers together (i.e., Transmission Control Protocol (TCP)/Internet Protocol (IP)/User Datagram Protocol (UDP), Bluetooth, ZigBee).

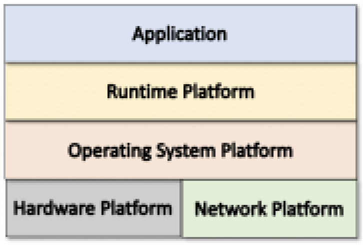


Figure 37: The Kinds of Platforms

As a consequence, in order to identify a specific platform, all the individual platforms and the specific versions need to be identified. This could result in hundreds if not thousands of combinations of platforms. This can quickly become a maintenance nightmare with versions that can almost change daily to apply patches.

```
HW-vvv:OS-vvv:RT-vvv
```

Where:

- HW - represents the specific hardware such as x86, 6800, ARM, etc
- OS - represents the specific Operating System such as Windows, MacOS, iOS, Android, etc
- RT - represents the specific runtime environment such as C, Java, Solidity, C#, eyc
- vvv - represents the specific version of the platform such as 10.15.7

- 4.2.1.1 Hardware Platform
- 4.2.1.2 Operating System Platform
- 4.2.1.3 Runtime Platforms
- 4.2.1.4 Network Platforms

Another way to represent a platform is to use an Application Container that encapsulates the platforms into the container. This simplifies the deployment and the number of user platforms that have to be supported. For example, any user platform that can support a container, will be able to deploy and use the Application.

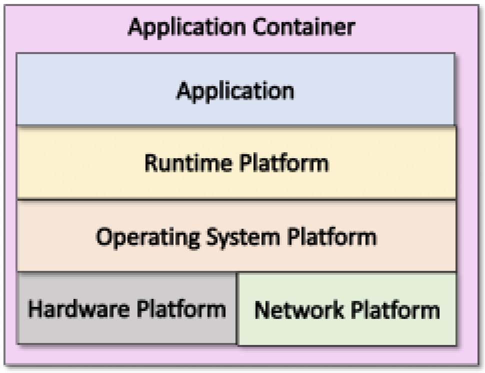


Figure 38: The composition of a Application Container.

- 4.2.1.5 Virtualized Nodes

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:1_func:platform

Last update: **2021/03/25 10:22**

4.2.1.1 Hardware Platform

Return to Platforms

About

In a Distributed System, each Network Node in the Node Network is associated with a Computer Architecture. As a consequence, a requirement of the distributed system is to identify specifically the computer architectures that are permitted.

It is not a simple task. For example, a coin DIDO might not initially consider a need to support embedded systems, however, the system might need to support Point-of-Sale operations, and as a general rule, these are implemented as embedded systems.

Another example, might be a supply chain DIDO. It might be obvious that this implementation requires embedded systems to read bar codes, RFIDs, and to monitor sensors used in production. As initially conceived, the system may not need Handheld Computers since the results of the embedded systems will be processed by Servers. However, often inventory is now being made available using general purpose smartphones that can even provide maps of where to find the item within the warehouse. Is it far fetched to believe that Amazon may need to employ a supercomputer to help analyze the 1.3B transactions a day? Imagine trying to play "what-if" games with that kind of data and getting responses back in a useful time frame. [150)]

- Embedded Systems
- Servers
- Desktops
- Handheld Computers
- Supercomputers
- Network Computers

[150)] Conga; What's Under the Hood Supporting 1.3B Transactions a Day? Salesforce by the Numbers, 31 October 2020, Accessed 8 December 2020, https://apttus.com/blog/salesforce-by-the-numbers/

4.2.1.2 Operating System Platform

Return to Platforms

On every Network Node in the Node Network is built using an Operating System (OS) Platform, therefore, and each instance of the Distributed Application (ĐApp or DApp) distributed throughout the Node Network must be created specifically for the operating system running on that node. Each build of the Dapp requires work, not just in the process of building the software, but also in terms of maintenance and support. Diversity in the number of OSs supported can drastically increase the cost of of the Dapp throughout the System Lifecycle of the Dapp. It also brings more problems when an OS is considered deprecated and is at End-of-life (EoL).

As a consequence, each Dapp must determine the Operating systems it will support: too few and it may adversely effect adoption, too many and the cost of maintenance may make the Dapp too costly to maintain.

One way to limit the number of operating systems is to select operating systems that are tailored to the specific environment. For example, what are the target environments for the Dapp:

- Embedded Processors?
- Mobile devices such as tablets and phones?
- Network devices such as Network Storage Devices (NDS) or Storage Area Networks?
- Enterprise Servers?
- Desktops?
- Workstations?

The following table lists most of the common OSs and the environments they support. It can be used to help provide a functional list of OSs required for the project.

Operating System	Embedded Systems	Handheld Devices	Desktops	Workstations	Enterprise	Network Devices
Android	X	X	X			
Apstra					X	X
Azure Real Time Operating System (or Azure RTOS)	X					
balenaOS	X					
Blackberry QNX	X	X				
CentOS			X	X	X	
Chromium OS			X			
Cisco Digital Network Architecture (Cisco DNA)						X
Cisco Internetwork Operating System (IOS)						X
Cisco IOS XR						X
Cisco NX-OS						X

Operating System	Embedded Systems	Handheld Devices	Desktops	Workstations	Enterprise	Network Devices
ClearOS					X	X
CloudReady						
ExtremeXOS						X
FreeBSD	X	X	X	X	X	X
FreeRTOS	X					
IBM i					X	
iOS		X				
Junos operating system (Junos OS)						X
LynxOS RTOS	X					
Nokia X Software Platform	X	X				X
Open Network Linux						X
OpenServer						X
Oracle Linux (OL)					X	
Oracle Solaris				X	X	
Red Hat Enterprise Linux (RHEL)					X	
SANtricity Software Operating System (OS)					X	X
SCO UnixWare					X	
SUSE Linux Enterprise Server (SLES)	X		X	X	X	
TrueNAS			X	X	X	X
Ubuntu Linux		X	X			
Windows Server						

4.2.1.3 Runtime Platforms

Return to Platforms

About

A **Runtime Platform** is the software components that are required to run application software that are not part of the Application itself, Hardware Platform or Operating Systems Platform. Some examples of Runtime Platform components are language specific libraries, or language specific virtual machines.

In general, an application is a sequence of steps or events directed and coordinated by the application. Some of the steps are actually contained within the application while others are executed by the Runtime Platform, Operating Systems Platform or the Hardware Platform on behalf of the application.[151].

When an application is translated into a target hardware instruction set by a compiler, there would be an extreme enlargement of the executable code if all the reused code from the software and hardware platforms were added to the executable. Alternatively, compilers often use compiler-specific external functions in pre-compiled code and assembled into runtime libraries that are linked during execution. These libraries are generally optimized for efficiency and for segregation of functionality for improved accuracy, prevention of common runtime errors, or security concerns.

Runtime libraries provide functionality by accessing the underlying operating or hardware platforms. For example, the use of floating point arithmetic or array/vector processing that use array processors or multiple cores. These considerations can often blur the boundary between standard application runtime libraries and language specific runtime libraries. Often compilers provide copies of runtime libraries optimized for the features of the OS and the hardware.

The concept of a runtime library should not be confused with an ordinary program library like those created by application programmers or delivered as third party such as Dynamic Link Library (.dll) or Shared Object (.so), meaning a program library linked at run time. For example, the programming language C requires only a minimal runtime library (commonly called `crt0`) but defines a large standard library (called `C standard library`) that each implementation delivers.

Consequently, the successful execution of an application depends on the proper versions and revisions of all other the Relocatable Objects. It also means that some kinds of errors can only be caught at execution or run time. These kinds of errors are best caught using strategic testing methodologies and strategies such as regression or unit testing (see: testability for more on testing).

[151] Mansourov, Nikolai and Campara, Djenana; 2011, Accessed: 10 December 2020, https://www.sciencedirect.com/topics/computer-science/runtime-platform

4.2.1.4 Network Platforms

Return to Platforms

A DIDO, by definition, is a collection of networked nodes. Traditionally, the network is assumed to be Ethernet with nodes connected over Local Area Network (LAN) and/or a Wide Area Network (WAN) using a Network Device.

The connections can either be:

- Wired Network using
  - Network Cabling
  - Universal Serial Bus (USB)
- Wireless Network using Wireless Fidelity (Wi-Fi)

Note: for Private, permissioned DIDOs, there may be explicit bans on wireless or even USB connections.

However, there is a lot of growth and development using other wireless connections other than Ethernet or USB. For example:

- Bluetooth
- ZigBee
- Near-Field-Communication (NFC)

Consequently, it is important to identify the kinds of connections that are required to support the DIDO. Some example are:

- Many contactless payments systems use Radio Frequency Identification (RFID) and NFC
- Many supply chains use RFID
- Many smart home efforts use WiFi and zigbee
- Many automobiles use Bluetooth to connect phones, make queries, play music, etc.

4.2.1.5 Virtualized Nodes

Return to Platforms

About

A Virtual Node can reside on any machine that supports either Virtual Machines (VMs) or Application Containers (or refered to as Containers). These can run many different Hardware Platforms and or Operating Systems (OS) Platforms. VMs are limited more by the hardware platforms they run on because of the footprint size of hypervisor, whereas containers are smaller and can consequently run on more machines.

Superficially, the two can appear almost identical. However, VMs have a full Host OS as well as a Guest OS while the Application Containers are lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. They are a bit less secure than VMs since Containers manage memory versus giving each instance its own “machine” complete with memory.

Containers and VMs have similar goals: to isolate an application and its dependencies into a self-contained unit that can run anywhere (i.e., Virtual Nodes)

Moreover, containers and VMs remove the need for physical hardware, allowing for more efficient use of computing resources through sharing, both in terms of energy consumption and cost effectiveness.

The main difference between containers and VMs is in their architectural approach.

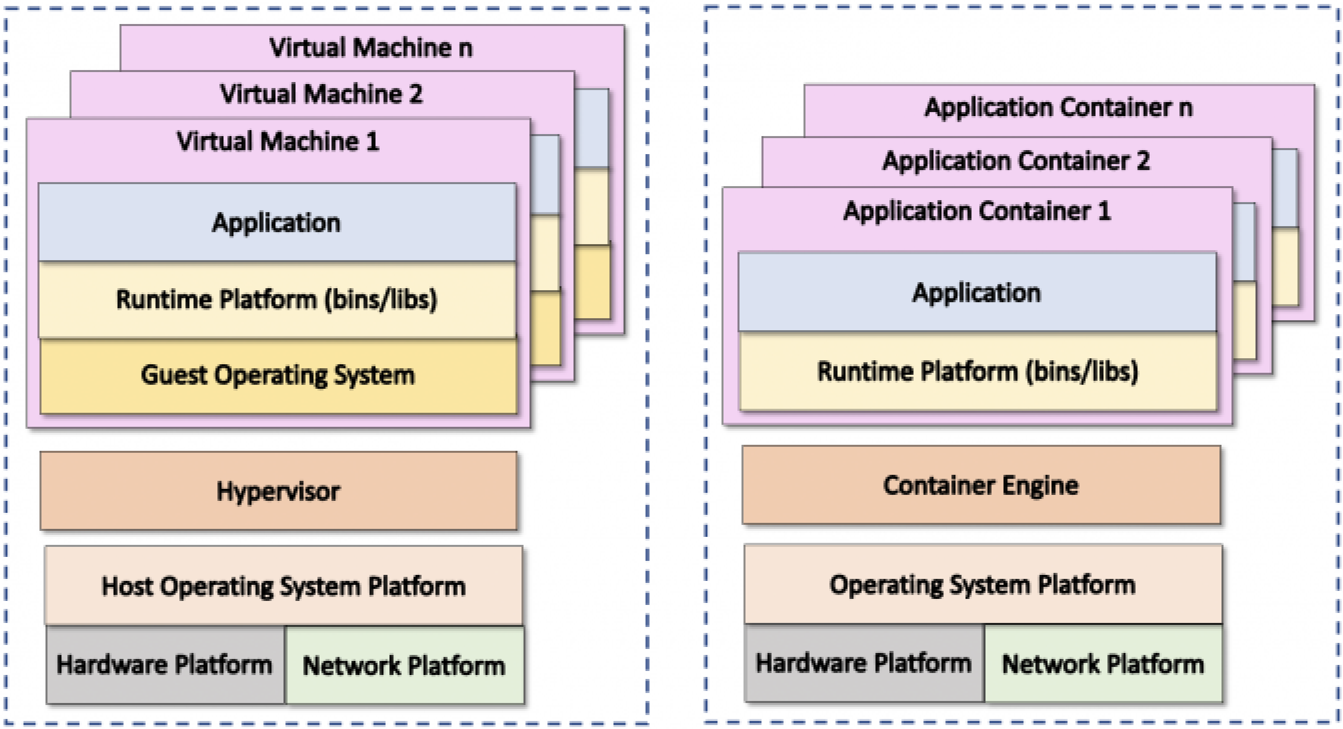


Figure 39: TVirtual Machines and Application Containers.

Virtual Machines

Return to Top

A Virtual Machine (VM) is a software program that behaves as if it were a complete computer. Often VMs are software versions of an existing Hardware Platform. For example, the Hardware Server running an Reduced Instruction Set Computer (RISC) processor, might host a VM that behave as if it were an x86, 68000, CISC, RISC, ARM, etc. The VM will have a virtual processor that executes the appropriate instruction set such as Complex Instruction Set Computer (CISC). Generally, the Hypervisor also known as a Virtual Machine Monitor (VMM) creates, runs and monitors the execution of the VMs. Each VM is isolated from each other and from the host machine which is a good for security reasons. For example, the memory (also virtual) is allocated for only for one VM. If another VM needs to use the same physical memory, the memory is “zeroed” before it can be reused.

As a general rule, Hypervisor's are heavy when compared to Container Engines and Virtual Machines are heavier than Containers. The “heaviness” is in reference to the resources required. For example, a Hypervisor takes more memory and Central Processing Unit (CPU) than a Container Engine because of the amount of work that is required to create and manage the VMs. This means that the smaller the host machine, the more likely that Containers are a better solution. Another reason that VMs are heavier is that each VM has its own copy of a a Guest Operating system while Containers are designed to share key parts of the host operating system.

Application Containers

Return to Top

An Application Container (also know as a Container) is a stand-alone, all-in-one package for a software application that run on an Container Engine. Containers include the the binaries and libraries used to run the application as well the support for the hardware needed to execute the application. Another way to think about a Container is that it is that everything need to execute the application is bundled into a single package (general one file).

Core Operating System (Core OS) is a system for container-based virtualization. Core OS deploys applications in virtual containers as a way to provide effective hardware virtualization for businesses.

- Container Engine
- Disk Image
- Virtual Disk Image (VDI)
- Virtual Machine Images

4.2.2 Access Control

Return to Functional Requirements

A major functional requirement is to provide a classifications of allowable nodes within the DIDO network. A detailed explanation of DIDO networks is provided by in Network Access Control.

Within each of these two classifications it is possible to have public and private access. Public and private access define who is able to write data onto a network or ledger. In contrast, open (i.e., permissionless) and closed (i.e., permissioned) determine who is able to read the data. Networks are classified as[152]:

- Permissionless Networks and Public Network - public and open
- Permissionless Networks and Private Network - public and closed
- Permissioned Networks and Public Network - private and open
- Permissioned Networks and Private Network - private and closed

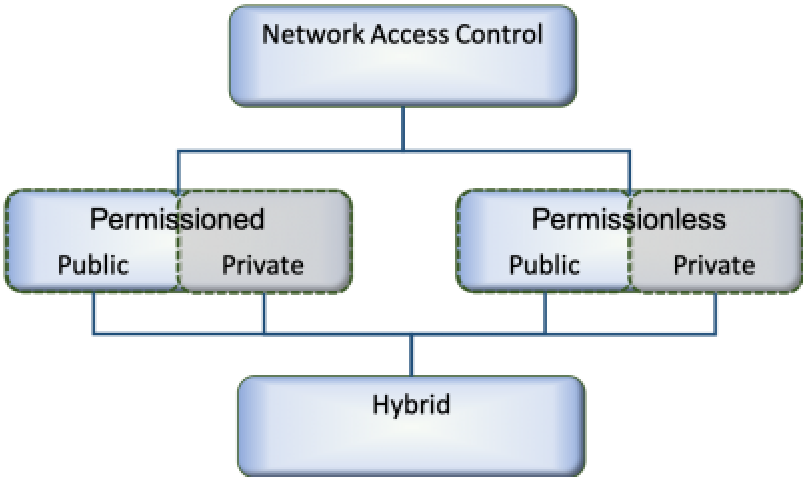


Figure 40: The Node Network Access Taxonomy

Determine the Access Control required for this DIDO by completing the worksheet presented in Table 4. Determine the characteristic required for the particular project of intrest. For example, for the Decentralization the answer should be either Permissioned or Permisionless. When the worksheet is done, use the answers to make the appropriate requirement of Permissionless versus Permissioned, public versus private, or if the requirements are a hybrid. Defining these requirements early can help avoid costly and time consuming changes later.

Table 4: Network Access Worksheet

Characteristic	Characteristic	Description
Decentralization		- **Permissionless** - Permissionless networks are decentralized and distributed. In other words, no one entity can close or terminate the network, modify the content, or censor parts of it. The larger the distributed and decentralized networks and or history are, the harder it is to tamper with.[153)] - **Permissioned** - The degree of decentralization for permissioned networks is a business decision. The extent and quality of decentralization depends upon the number of peers (i.e., nodes), the expected number of bad nodes in the network, and the type of consensus mechanism determined by the stakeholder. Permissioned blockchains usually employ an algorithm such as Byzantine Fault Tolerance, which differs from the proof of work (PoW) algorithm[154)].
Transparency		- **Permisionless** - Users or nodes have complete access to the ledger, transactions, and blocks in the blockchains, which allows for complete auditing of permissionless networks[155)]. - **Permissioned** - Transparency is not a driving force in permissioned networks and is often a major factor in the business decision to choose permissioned over permissionless networks. Most permissioned blockchains do not use cryptoeconomic coins incentive or tokens. The primary incentive of permissioned blockchain participants is to minimize the transparency, cost, time, and ease of sharing information3).
Privacy/Anonymity		- **Permisionless** - Privacy - In permissionless networks, users or nodes of the network are anonymized. Technically, permissionless networks like Bitcoin are pseudonymous, and not truly anonymous.[156)] - **Permissioned** - Anonymity - Permissioned blockchains offer fine-grained visibility into transaction details, as well as, metadata about those transactions which, in many ways, compromises the privacy of the Network participants[157)].

Characteristic	Characteristic	Description
Governance		• **Permisionless** - As a general rule, permissionless networks rely on open source software, which is ruled by open source communities (see Talk Openly Develop Openly (TODO)). The governance of the network is by consensus. Consensus is different for many of the permissionless networksm(i.e., Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA), etc). [158] • **Permissioned** - There are fundamental differences between permissionless and permissioned network governance. Permissioned governance is decided and agreed upon by members of the business network. Economic incentives, code quality, code changes, and power allocation among peers are based on the business dynamics and the common purpose and goals of the permissioned members. This allows for agile and responsive networks desired by businesses[159].
Tokens		• **Permisionless** - Permissionless blockchains employ fat protocols that compensate network contributors with Tokens. As the value and utility of the network increases, the value of the underlying tokens increases as well. This is the premise of cryptoeconomics and Initial Coin Offering (ICO) based fundraising. There are two predominant types of tokens today: monetary value tokens and utility tokens. Monetary value tokens are used in myriad ways as instruments for exchanging value. Utility tokens are akin to loyalty points: they have intrinsic value but no monetary value outside of that ecosystem.[160] • **Permissioned** - Permissioned blockchains generally do not employ a cryptoeconomic coins incentive or tokens[161].
Scalability & Performance		• **Permisionless** - For all the value blockchains bring to modern business processes, their Achilles heel often involves scalability and performance. Both Bitcoin and Ethereum blockchains suffer from poor scores in this area. For example, a recent blockchain game called Crypto kittles clogged the Ethereum network. Having said that, these are just early teething troubles, and startups are experimenting with various strategies to address this issue. Hopefully it is only a matter of time before this issue becomes a non-entity.[162] • **Permissioned** - Permissioned blockchains use consensus mechanisms, which are computationally inexpensive (when compared to proof of work (PoW)). Therefore, they enjoy substantially better scalability and performance than their permissionless network cousins[163].

Characteristic	Characteristic	Description
Open Read and Write		• **Public** - Anyone can participate by submitting transactions to the blockchain, such as Ethereum or Bitcoin; transactions can be viewed on the blockchain explorer.[164]
Ledger Is Distributed		• **Public** - The database is not centralized like in a client-server approach, and all nodes in the blockchain participate in the transaction validation.[165]
Immutable		• **Public** - When something is written to the blockchain, it can not be changed, in other words it is immutable.[166]
Secure Due to Mining		• **Public** - For example, with Bitcoin, obtaining a majority of network power could potentially enable massive double spending, and the ability to prevent transaction confirmations, in addition to other potentially malicious acts.[167]
Enterprise Permissioned		• **Private** - The enterprise controls the resources and access to the blockchain, hence private and/or permissioned.[168]
Faster Transactions		• **Private** - When you distribute the nodes locally, but also have far fewer nodes that participate in the ledger, performance is faster.[169]
Better Scalability		• **Private** - Being able to add nodes and services on demand can provide a great advantage to the enterprise.[170]
Compliance Support		• **Private** - As an enterprise, you would likely have compliance requirements to adhere to; having control of your infrastructure enhances ability to satisfy this requirement more seamlessly.[171]
Consensus More Efficient		• **Private** - Enterprise or private blockchains have fewer nodes and usually a different consensus algorithm, such as BFT vs PoW.[172]
Private Transactions		• **Hybrid** - Transaction are private but verifiable using the ledger's immutable data objects (i.e., leverage its public state). In its public state, each transaction gets approved by a massive network and is essentially secure and trustworthy. Hence, there is no need for a central governing body or an exhaustive chain of intermediaries to supervise things. So, any change done to a transaction will undergo a “kindred” approval process, making it next to impossible for a single actor to meddle with the transaction or entries[173].

Characteristic	Characteristic	Description
Equality		• **Hybrid** - Everyone in the network has equal rights to view, modify, and append their consent to a transaction. In addition, the identity of transacting parties is never disclosed to all the visible network participants. [174].
Non-Repuddiation		• **Hybrid** - Anonymity is simply not acceptable to financial institutions and regulated industries with their strict Know Your Customer (KYC) standards. [175].
Confidentiality		• **Hybrid** - Unrestricted visibility of the public state of the network exposes all the data to a colossal network breach, which is counter to data confidentiality obligations, as well as their business concerns.[176].

- **Note:** Another category of networks is a hybrid network, which makes it possible to restrict the visibility of information on the network using a combination of

public, private, permissionless and permissioned networks. Therefore, hybrid networks are appealing to regulated markets because they offer the benefits of public blockchain and private blockchain together.[177]

[152], [165], [166], [167], [168], [169], [170], [171], [172]
"Public Vs Private Blockchain In A Nutshell", Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

[153], [154], [155], [156], [157], [158], [159], [160], [161], [162], [163]
"Nuances Between Permissionless and Permissioned Blockchains", Anant Kadiyala, 18 February 2018, https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483

[164]
"Public Vs Private Blockchain In A Nutshell", Demiro Massessi, 12 December 2018, https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

[173]
"If you Thought Blockchain was Amazing, Wait till You Read about Hybrid Blockchain", Atul Khekade, 20 January 2018, https://www.entrepreneur.com/article/307794. This article uses the term "agnate approval" rather than "kindred approval"; however, agnate limits a kindred relationship to males only. Thus, we prefer the term "kindred" over "agnate"

[174], [175], [176]
"If you Thought Blockchain was Amazing, Wait till You Read about Hybrid Blockchain", Atul Khekade, 20 January 2018, https://www.entrepreneur.com/article/307794

[177]
"Hybrid Blockchain: Decentralized Option for Highly Regulated Markets - Few players in highly regulated markets have adopted blockchain technology. However, hybrid blockchain will change this.", Mina Down, 14 November 2018
https://blog.goodaudience.com/hybrid-blockchain-decentralize-highly-regulated-markets-900f30a37903

4.3 Non-Functional Requirements

Return to Requirements

About

Non-functional requirements are often incorrectly assumed rather than being explicitly defined by users. This can lead to problems towards the end of a project as the user expectations for non-functional requirements are not met. Many times, the developers dismiss non-functional requirements as non-testable and therefore not enforceable.

This lack of specificity in non-functional requirements sets the stage for conflicts between the users, system architects, systems engineers, and developers. For example, users expect software to start and run every time it is used however, the non-functional requirement of reliability may never have been explicitly specified.

Users expect new features to be added to a system and tested before they use them. Users assume the software is maintainable without an explicit declaration for “maintainability”. In many ways, they expect it to be an unwritten requirement and or goal. In other words, users expect the system to be analyzable, changeable, stable and testable[4]. For example, smartphone users will switch apps to other apps if the energy consumed by the app is not efficient. Efficiency is therefore a non-functional requirement. Energy consumption may also be a functional requirements (i.e., An application can not use more than 1040 mW (milli-Watt) per SMS message. [5].

The DIDO RA assumes the following non-functional requirements:

- 4.3.1 Portability
- 4.3.2 Reliability
- 4.3.3 Maintainability
- 4.3.4 Securability
- 4.3.5 Manageability
- 4.3.6 Usability
- 4.3.7 Performance
- 4.3.8 Interoperability
- 4.3.9 Elasticity
- 4.3.10 Scalability

Creating a Trade Study

Return to Top

A trade study or trade-off study helps consumers compare products on an equal footing, in other words, to help a consumer compare apples-to-apples so to speak. For example, when comparing cameras it is important to know the resolution metric of the camera. Simplistically, the higher the resolution, the better the camera. Speakers on the other hand might use the highest or lowest speaker frequency responses as a metric. Each metric is unique to the product being evaluated. However, if two cameras each have the same camera resolution, then the additional metric of frequency response might be used as a differentiator.

In the examples above, the camera resolution and frequency responses are specific to the product being evaluated and are referred to as functional requirements. However, there are also a set of requirements with corresponding metrics that capture products non-functional requirements (i.e., sometimes referred to as the "**ilities**" and include things like maintainab**ility**, portab**ility**, reliab**ility**, etc.)

Often a trade study is based on a collection of Figure of Merits with each FoM representing a single evaluation score for a single function. In the examples above, the camera resolution, or the high frequency response. An entire camera may have a FoM, but it represents the cumulative FoM of each function. A Camera may have other FoMs such as weight, battery life, size, or exchangeable lenses.

- How to Use the Boiler Plate

[4] Prolifics Testiing, Achieving Requirements Testability, 10 October 2018, Accessed 10 November 2020, https://www.prolifics-testing.com/news/achieving-requirements-testability

[5] Sai Suren Kumar Kasireddy and Vishnuvardhan Reddy Bojja, Measurements of EnergyConsumption in MobileApplications with respect toQuality of Experience, School of Computing, Blekinge Institute of Technology, 37179 Karlskrona, Sweden, March 2012, Acessed on 10 November 2020, https://www.diva-portal.org/smash/get/diva2:829733/FULLTEXT01.pdf

4.3.1 Portability

return to Non-Functional Requirements

About

Portability is the *degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub-characteristics:*[21)]

- ***Adaptability*** - *Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.*
- ***Installability*** - *Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.*
- ***Replaceability*** - *Degree to which a product can replace another specified software product for the same purpose in the same environment.*

The phrase “to port” means to modify software and make it adaptable to work on a different computer system. For example, to port an application to Linux means to modify the program so that it can be run in a Linux environment.

Portability also refers to the ability of an application to move across environments, not just across platforms. To clarify, a computer platform generally refers only to the operating system and computer hardware. A computer environment is much broader and may include the hardware, the operating system and the interfaces with other software, users and programmers.

[21)] ISO/IEC 25010, Portability, Accessed 27 July 2020, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/64-portability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:10_portability

Last update: **2021/06/09 13:35**

4.3.1.1 Adaptability

Return to Portability

About

Return to the Top

Adaptability is the degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments. In other words, it is the extent to which software systems adapts to changes in its environment such as operating system, databases, runtime environments, etc. An adaptable software system tolerates environmental changes without the need for external intervention. For example, a dual-mode cell phone can find out by itself if any one of the two wireless standards it supports is available at its current location and, if so, starts using that standard.[22)].

Nary Subramanian & Lawrence Chung[1] describe software architectures as composed of two elements: components and connectors. Adaptability in software can occur when either the components of the software change or the connectors between the software change.

Software Adaptability is when a software component with a well-defined, stable Application Programming Interface (API) can be exchanged using another component with minimal effort, as long as that component adheres to the API. For example, SQL describes an API for a database component. As long as the software adheres to the standard SQL API, the DataBase Management System (DBMS) can be exchanged between, for example, Oracle and PostgreSQL, with no to minimal impact.

Architecture Adaptability is when the connectors between software components change without having to change the components. This again comes down to having well-defined, stable APIs for the connectors. For example, the Unix File System (UnixFS) is a connector between software components and the physical filesystem. The associated UnixFS library can be exchanged for the InterPlanetary File System (IPFS) UnixFS connector and the software component should have no to minimal impact.

Nary Subramanian & Lawrence Chung[1] further define the following adaptability indices.

- **EAI** - **Element Adaptability Index** is a general purpose index, which represents a weighted value of adaptability where **1** represents complete adaptability and **0** represents no adaptability
- **AAI** - **Architecture Adaptability Index** is an EAI based on the number of elements in the architecture

$$AAI = \sum_{n=1}^{+\infty} EIA_{arc\ elements}$$

- **SAI** - **Software adaptability index** is an EAI based on the number of architectures for the software

$$SAI = \sum_{n=1}^{+\infty} EIA_{sw\,elements}$$

DIDO Specifics

Return to the Top

To be added/expanded in future revisions of the DIDO RA

=-
=-

[22] Nary Subramanian and Lawrence Chung, Metrics for Software Adaptability, University of Texas - Dallas, Accessed 29 July 2020, https://personal.utdallas.edu/~chung/ftp/sqm.pdf

4.3.1.2 Installability

Return to Portability

About

Installability and its mirror requirement of un-installability allows for an individual product or an entire system to be installed and uninstalled on a system easily, efficiently and with a minimum number of side effects. A side effect occurs when removing a product or system has adverse effects on other products or systems running on the same computer or network. For example, when a product was installed, it installed a library to access a particular DataBase Management System (DBMS) but, in so doing, removed a library required to access a different DBMS. These side effects can also occur when a product or system is un-installed. For example, when a product is un-installed, it removes a library used to access a particular DBMS; however, this library is required by other products or systems on the computer or network.

In addition, the complexity of installing or un-installing a product or system may be extremely tedious, detailed, and laborious with many steps and decision points. Each step in this process introduces risk. It is important to remember that risk is multiplicative, thus as the number of steps involved increases, even if the individual risk of each step is low, the overall risk to the success of the installation or un-installation increases.

Many of these issues were the bane of new systems and products in the past. Fortunately many of these issues have been addressed through the use of Wizards, Package Manager tools, containerization processes and orchestration tools. Installation usually entails the following checks and functions:

- The target system has the correct system resources available
  - Hardware architecture (32-Bit or 64-Bit Central Processing Unit (CPU) etc)
  - Operating System (OS) such as Android, IOS, Linux, MacOS, Windows, Unix, etc.)
  - Network connectivity
  - Hardware resources such as memory, disk space, etc.
- The target system does not already have the software installed (i.e., previous or current version of the software). If so, perform an upgrade.
- The target system has the proper directories, files and operating system privileges
- Add configuration data (i.e., configuration files, Environment Variables, or Windows Registry entries) to the target system
- Make software accessible on the target system (i.e., setting privileges, creating links, and shortcuts)
- Configure components required to run the target system (i.e., Daemon, services, etc)
- Activate software on the target system (i.e., license agreements, license activation, system registration, etc.)
- Update other third party components to acceptable levels on the target system

Installers can be classified according to the amount of interaction with the user:

Table 1: Installer Classifications

Kind of Installer	Description
Attended installation	This generally requires a user to attend the installation process to answer questions about where to do the install on the target system, provide information about the user, accept any terms and conditions, etc. [1]
Silent installation	This allows installation of a system or program on a target system without any notification to the user. This is often the backdoor used by malware. It differs from **Attended** and **Unattended** installations only in that the user may be completely unaware of the installation.
Unattended installation	This installation is similar to the **Attended Installation** but does not require any human interaction, thus allowing for systems or programs to be installed with the user just monitoring the installation rather than interacting with it.
Headless installation	This refers to the graphics head usually used to drive a monitor. If there is no graphics head, there are only command line interfaces and logs for detailing the installation process. Often these installations use Telnet where the installation on one computer (or machine) is being done from another computer (or machine). **Note:** could be a virtual machine.
Scheduled or automated installation	This term is usually used with **Attended** or **Unattended** installation. The installation process is scheduled for a later time, or on a schedule (i.e., every first Tuesday of the month).
Clean installation	This installation can be considered “hostile” in that no regard is made for previous installations. It can also mean that it does not care about other systems or programs that are installed. It `makes clean` and then starts the installation process anew.
Network installation	This kind of installation is made using a shared network resource. It often requires the installation of a minimal or skeleton operating system before the rest of the installation can occur. This kind of installation is used often for large corporate, government or other institutional organization.

[1] **Note:** It is possible to have hybrids of these kinds of installers. For example, an installer is **Attended** for the first part of an installation and then unattended for the remainder of the installation

In addition to the kinds of installations. Installers programs can either be **self contained**[1] and specific[2] in what they can install (i.e., **Wizard**) or they can be generalized to handle any kind of installation (i.e., **Package Manager**). Sometimes, installer programs themselves need to be installed or updated, a pattern referred to as "bootstrapping", which entails the use of a lightweight, simple and small executable file that is initially downloaded and started on the target system. This executable updates the installer software and then launches into the installation of the desired software. Often in complex systems or programs, the initial bootstrapping process updates other components that the desired software depends upon. For example, an operating system or DBMS update.

[1] **Note:** Contain all the files they need to perform the installation.

[2] **Note:** product based such as Microsoft Word, Parallels

DIDO Specifics

Return to the Top

To be added/expanded in future revisions of the DIDO RA

4.3.1.3 Replaceability

Return to Portability

About

Replaceability occurs when software components are developed using open, well written, standard specifications, usually captured as an Application Programming Interface (API). The standards can be either technical (i.e., developed by a Standards Organization) or de facto (i.e., developed by a for-profit or not-for-profit corporation). Replaceability is also a key factor in preventing Vendor Lock-In.

Replaceability is not just about the ability to switch suppliers and avoid Vendor Lock-In[182)] of the components, it's also about managing risk to the target system. This is especially true because each component can have its own Lifecycle with its own End-of-life (EoL) timelines, independent of the target system. In addition to the components' lifecycle, many components are now Open Source Software (OSS), which can often have forks spawning newer and competing products with similar, but not identical APIs. A recent article describes the Best Message Queue (MQ) Software of 2020. It describes 30 of the “top” Message Queue(MQ) Message-Oriented Middleware (MOM) software products.[183)]:

1. MuleSoft Anypoint Platform
2. IBM MQ
3. Azure Scheduler
4. Apache Kafka
5. Google Cloud Pub/Sub
6. Amazon MQ
7. RabbitMQ
8. Apache ActiveMQ
9. KubeMQ
10. IBM MQ on Cloud
11. Azure Queue Storage
12. Alibaba Message Queue
13. Alibaba Message Service
14. CloudAMQP
15. Intel MPI Library
16. ZeroMQ
17. Apache Qpid
18. Apache RocketMQ
19. IBM Compose for RabbitMQ
20. IronMQ
21. PubSub+
22. Red Hat AMQ
23. TIBCO Enterprise Message Service
24. Bottomline GTBridge
25. CloudMQTT

26. Compose Hosted RabbitMQ
27. deepestream.io
28. Enduro/Z
29. EnMasse
30. IBM Cloud Pak for Integration

Obviously, not all these products have the same API. They definitely do not have the same Wire Protocol, so it is very difficult to “mix and match” Publishers and Subscribers. In a distributed system, this would require all the components within the system to upgrade at the same time, sometimes referred to as a “Reboot the World Problem”. And when the upgrades occur, each part of the system requires complete Regression Testing to make sure that all the parts of the system continue to operate and function according to the specifications.

Replaceability is closely related to Adaptability and the kinds of Installers Architecture and Software Adaptability. Whenever there is an architecture or software adaptability issue, there is probably a Replaceability issue as well. Additionally, the concepts of Installability and Replaceability overlap. The harder a system or a product is to install, the greater the probability it will also be hard to replace.

Competitive products within the same domain are ideal candidates for Replaceability. However, replacing products should not just be driven by the cost of acquisition (i.e., purchase price) but also on the Total Cost of Ownership (TCO), which considers the cost throughout the lifecycle of the target system or component including maintenance (see 4.3.3 Maintainability). This cost can be tied to other tangential products such as debuggers, performance monitors, loggers, or any of the Non-Functional Requirements.[184]

DIDO Specifics

Return to the Top

To be added/expanded in future revisions of the DIDO RA

[182] **Note:** Vendors are not just proprietary corporations; Open Source projects produce and sell products also. The software might be “free”, but the producers are competitors that have the same drive to lock-in customers as the corporations

[183] Best Message Queue (MQ) Software, https://www.g2.com/,

[184] Portability Testing Guide with Practical Examples, Software Testing Help, 30 June 2020, Accessed 31 July 2020, https://www.softwaretestinghelp.com/what-is-portability-testing/

4.3.2 Reliability

return to Non-Functional Requirements

About

https://www.sebokwiki.org/wiki/Reliability,_Availability,_and_Maintainability#RAM_Considerations_during_Systems_Development

***Reliability** is the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub-characteristics:*[185]

- ***Maturity** - Degree to which a system, product or component meets needs for reliability under normal operation.*
- ***Availability** - Degree to which a system, product or component is operational and accessible when required for use.*
- ***Fault tolerance** - Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.*
- ***Recoverability** - Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.*

Reliability, maintainability, and availability (RAM) are three system attributes that are of great interest to systems engineers, logisticians, and users. Collectively, they affect both the utility and the life-cycle costs of a product or system. The origins of contemporary reliability engineering can be traced to World War II. The discipline's first concerns were electronic and mechanical components (Ebeling, 2010). However, current trends point to a dramatic rise in the number of industrial, military, and consumer products with integrated computing functions. Given the rapidly increasing integration of computers into products and systems used by consumers, industry, governments, and the military, reliability must consider both hardware and software.

Maintainability models present some interesting challenges. The time to repair an item is the sum of the time required for evacuation, diagnosis, assembly of resources (parts, bays, tool, and mechanics), repair, inspection, and return. Administrative delay (such as holidays) can also affect repair times. Often these sub-processes have a minimum time to complete that is not zero, resulting in the distribution used to model maintainability having a threshold parameter.

A threshold parameter is defined as the minimum probable time to repair. Estimation of maintainability can be further complicated by queuing effects, resulting in times to repair that are not independent. This dependency frequently makes the analytical solution of problems involving maintainability intractable and promotes the use of simulation to support analysis.

DDS Specifics

return to Top

To be added/expanded in future revisions of the DIDO RA

[185] ISO/IEC 25010, Reliability, Accessed 27 July 2020, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/62-reliability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:14_reliability

Last update: **2021/06/11 14:52**

4.3.2.1 Maturity

Return to Glossary

About

Return to Top

There are two ways to think about maturity: the maturity of the products or systems and the maturity of the communities which develop the systems or products. Usually, the two kinds of maturity go hand in hand. A mature product or system is the result of a mature community process and, visa versa, a mature community process produces mature products.

- **Note:** Community, in this case, can refer to a Community of Interest (CoI) or a corporation.

Products or Systems

Return to Top

Product or System maturity is an assessment (sometimes quantifiable) of how well a product or system meets its requirements for reliability under normal operations.

Maturity of the components selected for inclusion in a system can play a significant role in the overall success of a system. Components that are mature are more likely to be stable and reliable; qualities that directly translate to stable and reliable integrations, which are thereby robust and resilient when inevitable changes to the system are made. This holds true as long as the components are not coming close to End-of-life (EoL). See Manageability Costs .

Rafa E. Al-Qutaish and Alain Abran have proposed a maturity model based on Six Sigma (6Sigma).[186)]

> *The quality of a product can be assessed either directly by looking into the product itself, or indirectly through assessing the process used to develop that product. In the software engineering field, there are currently numerous capability and maturity models for assessing a set of specific software processes, but very few product maturity models for those interested in assessing the quality of software products. This paper presents a maturity model designed to directly assess the quality of a software product, i.e. the Software Product Quality Maturity Model (SPQMM). This model is based on the six-sigma view of product quality and handles – in submodels – the three views of quality specified in ISO 9126, that is, the Software Product Internal Quality Maturity Model (SPIQMM), the Software Product External Quality Maturity Model (SPEQMM), and the Software Product Quality-in-Use Maturity Model (SPQiUMM).*

$$WQL = \frac{\left(IQL + EQL + iUQL\right)}{3}$$

Where:

- ***WQL*** *is the quality level of the whole software product, including the quality levels of all three stages of the software product*
- ***IQL*** *is the internal quality level*
- ***EQL*** *is the external quality level*
- ***iUQL*** *is the in-use quality level of the software product*

Table 6: Sigma values based on the quality level and the software integrity level

5	**4**	**3**	**2**	**1**	**0**	
Very High	**High**	**Intermediate**	**Low**	**Trivial**	**None**	
Zero Sigma Shift	1.5 Sigma Shift	2.0 Sigma Shift	2.5 Sigma Shift	3.0 Sigma Shift	3.5 Sigma Shift	Assigned Sigma Ranges
QL ≥ 99.99997%	QL ≥ 99.976%	QL ≥ 99.865%	QL ≥ 99.379%	QL ≥ 97.724%	QL ≥ 03.319%	σ ≥ 5
QL < 99.99997%	QL < 99.976%	QL < 99.865%	QL < 99.379%	QL < 97.724%	QL < 93.319%	
and	and	and	and	and	and	5 > σ ≥ 4
QL ≥ 99.996%	QL ≥ 99.379%	QL ≥ 99.724%	QL ≥ 99.319%	QL ≥ 97.134%	QL ≥ 69.146%	
QL < 99.996%	QL < 99.379%	QL < 97.724%	QL < 93.319%	QL < 84.134%	QL < 69.146%	
and	and	and	and	and	and	4 > σ ≥ 3
QL ≥ 99.865%	QL ≥ 93.319%	QL ≥ 84.134%	QL ≥ 69.146%	QL ≥ 50%	QL ≥ 30.853%	
QL < 99.965%	QL < 93.319%	QL < 84.134%	QL < 69.146%	QL < 50%	QL < 30.853%	
and	and	and	and	and	and	3> σ ≥ 2
QL ≥ 97.724%	QL ≥ 69.146%	QL ≥ 50%	QL ≥ 30.853%	QL ≥ 15.865%	QL ≥ 6.680%	
QL < 97.724%	QL < 69.146%	QL < 50%	QL < 30.853%	QL < 15.865%	QL < 6.680%	σ < 2

Communities

Return to Top

There are several ways to establish or assess Community[187] maturity:

- ISO 9001
- ISO 15288
- ISO 90003-2018
- ISO 10001:2018 Quality management — Customer satisfaction — Guidelines for codes of conduct for organizations

- ISO 10002:2018 Quality management — Customer satisfaction — Guidelines for complaints handling in organizations
- ISO 10003:2018 Quality management — Customer satisfaction — Guidelines for dispute resolution external to organizations
- ISO 10004:2018 Quality management — Customer satisfaction — Guidelines for monitoring and measuring
- OMG: Case Management Model and Notation (CMMN)
- Capability Maturity Model Integration (CMMI)

See also: Talk Openly Develop Openly (TODO).

DIDO Specifics

Return to Top

[186] Rafa E. Al-Qutaish and Alain Abran, A Maturity Model of Software Product Quality, Journal of Research and Practice in Information Technology, 43(4):307-327, November 2011, Accessed 27 July 2020, https://www.researchgate.net/publication/260835325_A_Maturity_Model_of_Software_Product_Quality

[187] **Note:** Community in this case can refer to a Community of Interest (CoI) or a corporation.

4.3.2.2 Availability

Return to Reliability

About

return to the top

Availability in this context is System Availability. System Availability is the probability the system will function as designed for a particular duration. The duration could be a fixed time period (i.e., 24 hours a day, 7 days a week, or 364.9 days a year) or it could be over a particular mission (i.e., a flying mission, a patient stay, or a growing season). The ability of an item to be in a state to perform a required function under given conditions at a given instant of time or over a time interval, assuming the required external resources are provided. [188)]

It is important to remember that Availability is expressed as a probability expressed in terms of Five Nines, and therefore, the Multiplication Rule of Probability needs to be considered when thinking about a system comprised of parts. Each part has its own probability of success (or failure). The **Multiplication Rule of Probability** means that to find the probability of the intersection of two events, multiply the two probabilities. The intersection of the events occurs when the probability of two events occurring is known. The Multiplication Rule of Probability determines the intersection of two different sets of events, called independent and dependent events.

- An Independent Event is when the probability of an event is not affected by a previous event.
- A Dependent Event is when one event influences the outcome of another event in a probability scenario. To find the intersection of two events, whether they are independent or dependent, multiply the two probabilities together. [189)]

$$A_o = \frac{T_m}{T_m + T_d} \begin{cases} A_o = Operational\ Availability \\ T_m = Mission\ Duration \\ T_d = Observed\ Down\ Time \end{cases}$$

Mission Duration (T_m) is the time the system needs to be operational. T_m can be expressed as a fixed time period (i.e., 24 hours a day, 7 days a week, or 365 days a year) or it could be over a particular mission (i.e., a flying mission, a patient stay, peak energy demand, or a growing season)

Mean Time Between Failure (MTBF) is a calculation of the arithmetic mean (average) time between failures of a system.

- **Note:** If a system is designed with both redundancy and automatic fault bypass, then MTBF is the anticipated lifespan of the system if these features cover all possible failure modes (infinity for all practical purposes). Such systems will continue without noticeable interruption when these conditions are satisfied unless there are secondary failures. This is called active redundancy and requires no maintenance to prevent mission failure. Active redundancy is required for systems that cannot be maintained, such as satellites. See: Wikipedia, Mean Time Between Failure (MTBF), Accessed 3 July 2020, https://en.wikipedia.org/wiki/Availability_(system)

- **Note:** The term is used for repairable systems[190]

Downtime (T_d**)** is the Mission Duration times the sum of all of the different kinds of time required to transition from being down to the time to be fully operational, divided by the Mean Time Between Failure.

- Mean Time To Repair (MTTR) is the time required to restore operations the level defined in the system specification
- Mean Logistics Delay Time (MLDT) is the time required to obtain parts from the part depot or from the manufacturer including transportation to the site
- Mean Active Maintenance Down Time (MAMDT) is the average time required to perform diagnostics and replace parts

DIDO Specifics

return to the top

[188] **Note:** Availability is part of Reliability, Maintainability, and Availability (RAM)

[189] The Multiplication Rule of Probability: Definition & Examples, Chapter 4, lesson 11, Accessed 3 July 2020, https://study.com/academy/lesson/the-multiplication-rule-of-probability-definition-examples-quiz.html

[190] https://en.wikipedia.org/wiki/Mean_time_between_failures | Mean Time Between Failure (MTBF)]]

4.3.2.3 Fault Tolerance

Return to Reliability

About

Return to the Top

Fault Tolerance is the ability of a system (computer, network, cloud cluster, component, etc.) to continue functioning correctly without interruption during failures. Fault Tolerant systems (or components) prevent disruptions to a system that is considered Safety-Critical System (SCS), Life-Critical System or Mission Critical System. Usually, this requires an understanding of the single points of failure through the multiple critical execution paths in a running system.

The system characteristics of Fault Tolerance High Availability are related in that to achieve high availability, a system must address Fault Tolerance of components on the systems critical paths.

Fault Tolerant systems use redundant (i.e;, spare, backup) components to automatically become available in the event of a component failure to ensure there is no loss of service or data. The ability to use Failover mechanisms to quickly, smoothly and transparently transition to the redundant or backup systems requires a well designed system, with contingency plans and special management processes, hardware or software to ensure the transition. There are some Failover components which are acquired. For example:

- **Power sources** are ruggedized as fault tolerant by incorporating alternative sources and backups like Uninterruptible Power Supply (UPS) and backup generators. A good description of this is provided in the Tactical Microgrid Standard (TMS) use case,
- **Hardware systems** are made Fault Tolerant by deploying identical or equivalent systems that can either be used instead of the original system or use in conjunctions with the original system used as an alternative. For example, a server can be made fault tolerant by using an identical server running in parallel, with all operations mirrored to the backup server.
- **Networks** designed as Fault Tolerant by supporting multiple networks paths between any two endpoints within the Local Area Network (LAN) or Wide Area Network (WAN) are possible but the actual endpoint also needs to be duplicated (i.e., two Network Interface Cards (NICs)). It is also possible to use two different networks such as a wired, Wireless Fidelity (Wi-Fi), Bluetooth, or ZigBee.
- **Software** systems or components become fault tolerant when multiple instances of the software are running in parallel using either operating system threads or even more modern containers such as Docker or orchestration software such as Kubernetes. For example, a database can be continuously replicated to other machines. If the primary database goes down, operations can be automatically redirected to the second database. Another example, would be use of orchestration software such as Kubernetes to automatically use an alternate application container on the same or different machine.

Fault Tolerance needs to be considered in all disaster recovery plans or strategies. For example, Fault

Tolerant systems can use the cloud for backups allowing critical systems to quickly be restored. Although these backups are not true immediate failovers, they can offer a longer timeline for fault tolerance recovery. **Note:** often these backup plans are not geographically local which is particularly important during natural or even human disasters. [191]

DIDO Specifics

Return to the Top

[191] Mariah Timms, AT&T outage: Internet, 911 disrupted, planes grounded after Nashville explosion. Get the latest updates, Nashville Tennessean, 5 January 2012, Accessed: 8 January 2021, https://www.tennessean.com/story/news/local/2020/12/25/att-outage-internet-down-hours-after-nashville-explosion/4045278001/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:14_reliability:04_faulttolerance

Last update: **2021/03/25 10:28**

4.3.2.4 Recoverability

Return to Reliability

About

Return to the Top

Recoverability is the ability of a system to be rebuilt in the event of a system failure do to human or natural disasters or catastrophic failures in hardware or software. After the system is recovered it is able to resume with full functionality with minimum interruption. For example in DataBase Management System's (DBMS), a Checkpoint is a place in time where the database transactions, operations, and logging are paused long enough to be completed and recorded into the database files. These files are then archived (i.e., copied, backed up) away from the current database files. After the Checkpoints operation is complete, the DBMS can resume with new transactions, operations and logging. Although the concepts of Checkpoints are usually thought of in conjunction with DBMSs, it is also possible to have Checkpoints applied to Operating Systems (OSs) as well. Virtual Machines (VMs) and containers (i.e., Docker) can be thought of as a checkpoint made against the OS files, logs, etc. at a particular point in time. Every time the VM or Container are reloaded, they start from a set known point (a Checkpoint).

DIDO Specifics

Return to the Top

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:14_reliability:12_recoverability

Last update: **2021/03/25 10:28**

4.3.3 Maintainability

return to Non-Functional Requirements

About

Maintainability is the *characteristic that represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in its environment and requirements. This characteristic is composed of the following sub-characteristics:*[192]

- ***Modularity*** *- Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.*
- ***Reusability*** *- Degree to which an asset can be used in more than one system, or in building other assets.*
- ***Analysability*** *- Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.*
- ***Modifiability*** *- Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.*
- ***Testability*** *- Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.*

Maintainability is a characteristic of a system that is able to resume full operation after a failure in a component of the system. Components that are mission critical for the system can include equipment, machine, power, air conditioning, software, etc. Maintainability is expressed as the probability of recovery based on a specified time frame, usually done in terms of Five Nines (i.e., 99%, 99.9%, 99.99%, and 99.999%). For example, a 99.999% maintainability would be for 5 minutes, 15 seconds or less of downtime in a year. The downtime must include all the steps required to recover with full operational capabilities, so the time must include removal, diagnostics, assembly of resources required to perform the maintenance (i.e., parts, bays, tools, personnel, etc.) and the re-installation of the failed component.[193] [194]

Two main aspects of Maintainability need to be addressed for all systems or components:

- Servicability - the ease of conducting scheduled inspections and servicing
- Repairability - the ease of restoring service after a failure

Maintainability is a projection of the downtime of a system. Given that it is, by its very nature, a projection, it should not viewed as a guarantee that a system will only be down for the projected amount of time. Maintainability must therefore rely on models to calculate the probability of failures for components based on actual failure rates for components in the past or test results of the components.

> *There are a wide range of models that estimate and predict reliability (Meeker and Escobar 1998).*

> *Simple models, such as exponential distribution, can be useful for “back of the envelope” calculations.*
>
> *System models are used to:*
>
> > *(1) combine probabilities or their surrogates, failure rates and restoration times, at the component level to find a system level probability or*
> > *(2) to evaluate a system for maintainability, single points of failure, and failure propagation. The three most common are reliability block diagrams, fault trees, and failure modes and effects analyses.*
>
> *There are more sophisticated probability models used for life data analysis. These are best characterized by their failure rate behavior, which is defined as the probability that a unit fails in the next small interval of time, given it has lived until the beginning of the interval, and divided by the length of the interval.* See: Upkeep's discussion of models in Maintainability Definition & Calculation [195]

All these models are abstractions of reality; therefore, at best they are only approximations of reality. To the extent they provide useful insights, they are still very valuable. The more complicated the model, the more data necessary to develop precise estimations. The greater the extrapolation required for a prediction, the greater the imprecision. Also, obtaining all the data required as input to the models is difficult, time consuming, and may not even be very accurate.

Measuring the Mean Time To Repair (MTTR), is also used as part of Availability (see 4.3.2.2 Availability).

The MTTR, identifies the average time to restore a system or component after experiencing a failure or breakdown in the expected (i.e., specified) operating conditions. The formula for MTTR is:

$$MTTR = \frac{Total\,downtime\left(hours\right)}{Number\,of\,failure\,events}$$

A lower MTTR value corresponds to a higher level of maintainability and which means that maintainable systems take less time to repair.

DIDO Specifics

Return to Top

All systems, regardless of their level of complexity, require maintenance. To reduce the impact of performing maintenance, using physical (hardware) modules (components) that require the fewest number of repairs in a given time frame and choosing hardware and designs that require the least amount of downtime is one way to reduce the impact of maintenance. Another way to reduce the impact of maintenance is to design a system that anticipates maintenance and provides redundancy to handle the downtime required for maintenance. Nevertheless, managing redundancy is not a trivial task and adds complexity to the overall system. The more components that require redundancy, the more complex the system becomes unless the Middleware can manage the transition seamlessly, easily and transparently.

There are a few major forms of redundancy DDS can help with:

Hardware redundancy	Means that there are multiple modules (components) that provide the same functionality available in the system at the same time. For many systems, a simple dual modular redundancy is sufficient to accomplish the job (i.e., two components). However, for life critical systems, some systems rely on a triple modular redundancy. These systems should have zero to minimum loss from downtime.
Information redundancy	Occurs when there are multiple information sources available, so that modules can use whichever source is active if there is an interruption in information. In a system that has only one network available to it, the information redundancy is of little use if the network needs maintenance.

[192] ISO/IEC 25010, Maintainability, Accessed 27 July 2020, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/64-maintainability

[193] Maintainability Definition & Calculations, Upkeep, answered 4 June 2019, Accessed 13 July 2020, https://www.onupkeep.com/answers/preventive-maintenance/maintainability-definitions-calculations/

[194] **Note:** Maintainability is part of Reliability, Maintainability, and Availability (RAM)

[195] Maintainability Definition & Calculations, Upkeep, answered 4 June 2019, Accessed 13 July 2020, https://www.sebokwiki.org/wiki/Reliability,_Availability,_and_Maintainability#Models

4.3.3.1 Modularity

Return to Maintainability

About

Modularity describes a characteristic of a system or program to be organized into smaller, reusable components. Each component is self-contained and provides an interface that describes the functionality it offers to other components of the system. It optionally provides a set of interfaces required to fulfill its functionality. In Object-Oriented Programming (OOP), this functionality is encapsulated as set of data attributes. The Module exposes access to these data attributes by defining public interfaces other components can call to manage and manipulate the data attributes of the object (i.e., Module). If the component relies on other components, then it can specify the required components (i.e., Modules).

Many Modules are described by files that only describe the interface to the Module and contain no actual functionality. For example, in C/C++, these Module interfaces are described using header files (i.e., `.h`, `.hpp`) files; in Java these files are regular `.java` files but contain the key word `interface` in their class descriptor; in Common Object Request Broker Architecture (CORBA), interfaces are described as stubs. ECMAScript (i.e., JavaScript), PHP, etc. use the concept of Duck Typing at runtime to accomplish the equivalent of interfaces (i.e., it is based on the methods defined within a Module at runtime rather than on a static, abstract fixed interface).

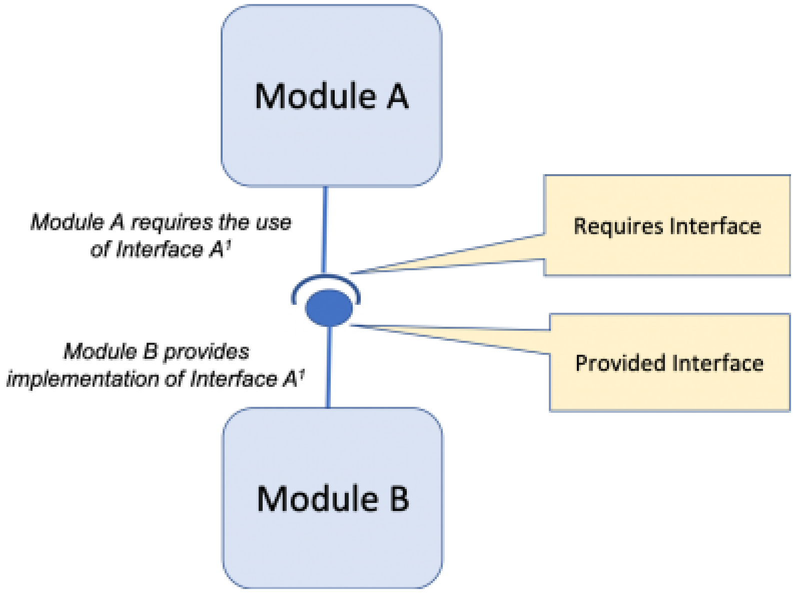

Module A
Module A requires the use of Interface A¹
Requires Interface
Provided Interface
Module B provides implementation of Interface A¹
Module B

Figure 41: Modeling Modules with Provided and Required Interfaces.

Regardless of the kind of implementation (i.e., interface, stubs, DuckTyping, etc.), APIs need to be public, formalized and freely available to the general public at no cost. The benefit of using interfaces, stubs, or headers is that un-implemented or mismatched functionality can be checked at compile or link time. Dynamic Plug In interfaces (i.e, DuckTyping) can only be checked at runtime. However, dynamic type is very definitely modular. One way around run-time errors is to always provide a default implementation.

The use of Modules allows the architecture and design of a system or program to be re-factored by extracting functionality into new Modules, combining Modules into a new Module, or composing Modules from other Modules. Often the rules for refactoring a system of a program follow many of the same rules as data Normalization.

Yiming et al.[196] introduced the use of complex network theory into software engineering with which to develop metrics for measuring Modularity. They analyzed existing software by representing it as a network using a feature coupling network (FCN). Each method and attribute is considered a node in the node network. Couplings between methods and attributes are considered edges. Edge Weight represents a weighted value based on the number of invocation paths for the node (i.e., method 'A' is called 2 times, has a weight of '2', each attribute is used once, gets a weight of '1').

`FCN = ( N, E, Ψ )`
Where:

- FCN : Feature Coupling Network
- N : the Node set (number of attributes or methods)
- E : the Edge set (couplings between Edge Nodes) of
- Ψ : the Edge Weight (number of invocation paths)

Yiming et al., apply Weyuker’s criteria, which are widely used in the field of software metrics, to validate modularity as a software metric theoretically, and also to perform an empirical evaluation using open-source Java software systems to show its effectiveness as a software metric to measure software modularity.

DIDO Specifics

Return to Top

Although Modularity was primarily developed to look at software, it can also be applied to distributed systems by changing the definition of N from the number of attributes or methods to the number of Publishers and Subscribers, and E to the number of couplings between the publishers and subscribers. Ψ would then be a weighting for the volume of couplings between publishers and subscribers.

In practice this means that if functionality is centralized into a normalized set of Endpoints such that couplings become the connections between endpoints and the volume of traffic represents a weighting, the end result is that we are reduced to three variables: N, E and Ψ. Thus, we must address the age old question: is it better to have one command with 100 options, 100 commands with no options, or a set of commands that can share a restricted set of options? It's obvious that the first two choices are unacceptable so we are left with with the last choice where the options are all related based on the functionality of the command.

[196] Yiming Xiang, Weifeng Pan, Haibo Jiang, Yunfang Zhu, Hao Li, Measuring Software Modularity Based on Software Networks, 14 February 2019, Entropy, Accessed 3 Aug 2020, https://www.mdpi.com/1099-4300/21/4/344/htm

4.3.3.2 Reusability

Return to Maintainability

About

Reusability is a way to save time, resources and effort by reusing existing system or software assets that were created previously. Sometimes these assets maintain all the functionality they were originally constructed to provide (e.g., widgets in a Graphical User Interface (GUI)). Sometimes, well written modularized software is abstracted into generalized Modules that can then be used to serve one or more design patterns, e.g., a linked list, hash algorithm, etc.) When this is done, the software is considered to be reusable.

Reusability is defined as the *ability* of something to be used more than once. In contrast, *reuse* is defined as the *action* of using something more than once. Reusability promotes and enables reuse but does not ensure complete reusability.[197]. Therefore, merely creating a system or program for Reusability is not important unless there is also a "marketing effort" to promote the reuse of the system or product. Marketing in this context does not mean commercial marketing. Often marketing can be achieved just through making the code available as Open Source.

The meaning and application of 'reuse' also depends on who you ask. To a Software Engineer, they naturally think about reuse of code, executables or even software patterns, etc. To a Systems Engineer, reuse makes them think about use cases, state transition diagrams, etc. To an Ontologist, reuse means glossaries, vocabularies and ontologies. To a contracting officer, they makes them think of contracts and performance specifications. Thus, it is obvious that even within a technology area, there are lots of different ideas about what, where, and how to re-use artifacts.

Jones[198] identifies 10 different areas within a software project where reuse can be applied:

1. Architectures
2. Source Code
3. Data
4. Designs
5. Documentation
6. Estimates
7. Human Interfaces
8. Plans
9. Requirements
10. Test Cases

Types of Reuse

Return to Top

Reuse is a multi-faceted idea, which requires an overall classification for the various types of reuse. Frakes and Terry [199] have developed a taxonomy for reuse based on an extensive review of the literature on Reuse and Reusability. Table 7 depicts the two major elements of the reuse taxonomy: **Facets**, which cover all the types of reuse, and **Terms**, which are used to describe each **Facet**. For example, the **Development Scope Facet** has two possible **Terms** within it: **Internal** and **External**. When describing the **Development Scoping** used in a company or project, this can be **Internal**, **External** or both. The **Terms** associated with each **Facet** are defined in Table 8.

Table 7: Types of Software Reuse [3]

Facet					
Development Scope[1]	**Modification**[2]	**Approach**[3]	**Domain Scope**[4]	**Management**[5]	**Reused Entity**[6]
Internal (Private)	White Box	Generative	Vertical	Systematic (Planned)	Code
External (Public)	Black Box (vrbatum)	Compositional	Horizontal	Ad Hoc	Abstract Level
	Adaptive (porting)	In-the-Small			Instance Level
		In-The-Large			Customization Reuse
		Indirect			Generic
		Direct			Source Code
		Carried Over			
		Leveraged			

[1] **Development Scope** refers to whether the reusable components are from a source external or internal to a project.
[2] **Modification** refers to how much a reusable asset is changed.
[3] **Approach** refers to different technical methods for implementing reuse.
[4] **Domain Scope** refers to whether reuse occurs within a family of systems or between families of systems.
[5] **Management** refers to the degree to which reuse is done systematically.
[6] **Reused Entity** refers to the type of the reused object.

Clear definitions of types of reuse are necessary prerequisites to measurement. Table 2 provides a faceted classification of reuse definitions gathered from the literature Each column specifies a facet, with the facet name inbold.

Table 8: Definitions of Types of Reuse [3]

Type of Reuse	**Description**
abstract-level	Abstract-level reuse is the use of high-level abstractions within an object-oriented inheritance structure as the foundation for new ideas or additional classification schemes.
ad-hoc	Ad-hoc reuse refers to the selection of components that are not designed for reuse from general libraries or where reuse is conducted by an individual in an informal manner

Type of Reuse	Description
adaptive	Adaptive reuse is a reuse strategy that uses large software structures as invariants and restricts variability to low-level, isolated locations. An example is changing arguments to parameterized modules.
black-box	Black-box reuse is the reuse of software components without any modification. See **verbatim**.
Carry-Over	Carry-Over Reuse is when software used with one version of a software component is carried over and used as is in a subsequent version of the same system.
compositional	Compositional reuse is a reuse strategy that uses small parts as invariants and then uses variant functionality to link these parts together. Programming in a high level language is an example. Compositional reuse is the use of existing components as building blocks for new systems. The Unix shell is an example
customization	Customization reuse is the use of object-oriented inheritance to support incremental development. A new application may inherit information from an existing class, overriding some methods in that class and adding new behaviors.
direct	Direct reuse is reuse without going through an intermediate entity.
external	External reuse level is the number of lower level items from an external repository in a higher level item divided by the total number of lower level items in the higher level item. See **Public**.
generative	Generative reuse is reuse at the specification level using application or code generators. Generative reuse offers the “highest potential payoff.” The Refine and MetaTool systems are state of the art examples.
generic	Generic reuse is reuse of generic packages, such as templates for packages or subprograms.
horizontal	Horizontal scope reuse is reuse of generic parts in different applications. Booch Ada Parts and other subroutine libraries are examples.
In-the-large	Reuse-in-the-large is the use of large, self-contained packages such as spreadsheets and operating systems.
In-the-small	Reuse-in-the-small is the reuse of components that are dependent on the environment of the application to achieve full functionality. Favaro asserts that component-oriented reuse is reuse-in-the-small.
indirect	Indirect reuse is reuse through an intermediate entity. The level of indirection is the number of intermediate entities between the reusing item and the item being reused
instance-level	Instance-level reuse is the most common form of reuse in an object-oriented environment. It is defined as simply creating an instance of an existing class.
internal	Internal reuse level is the number of lower level items not derived from an external repository and used more than once divided by the total number of lower level items not derived from an external repository. See **Private**.
leveraged	Leveraged reuse is reuse with modifications.
private	Private reuse is“the extent to which modules within a product are reused within the same product.” See **Internal** .
public	Public reuse is “the proportion of a product which was constructed externally.” See **External**.
source-code	Source code reuse is the low-level modification of an existing object-oriented class in order to change its performance characteristics.

Type of Reuse	Description
systematic (planned mode)	Systematic/planned mode reuse is the systematic and formal practice of reuse as found in software factories.
verbatim	Verbatim reuse is reuse of some item without modifications. See **Black-Box**.
vertical scope	Vertical scope reuse is reuse within the same application or domain. An example is domain analysis or domain modeling.
white-box	White-box reuse is the reuse of components by modification and adaptation.

Types of Metrics and Models

Return to Top

Frakes and Terry categorize the kinds of metrics and models that can be used for evaluating reuse.

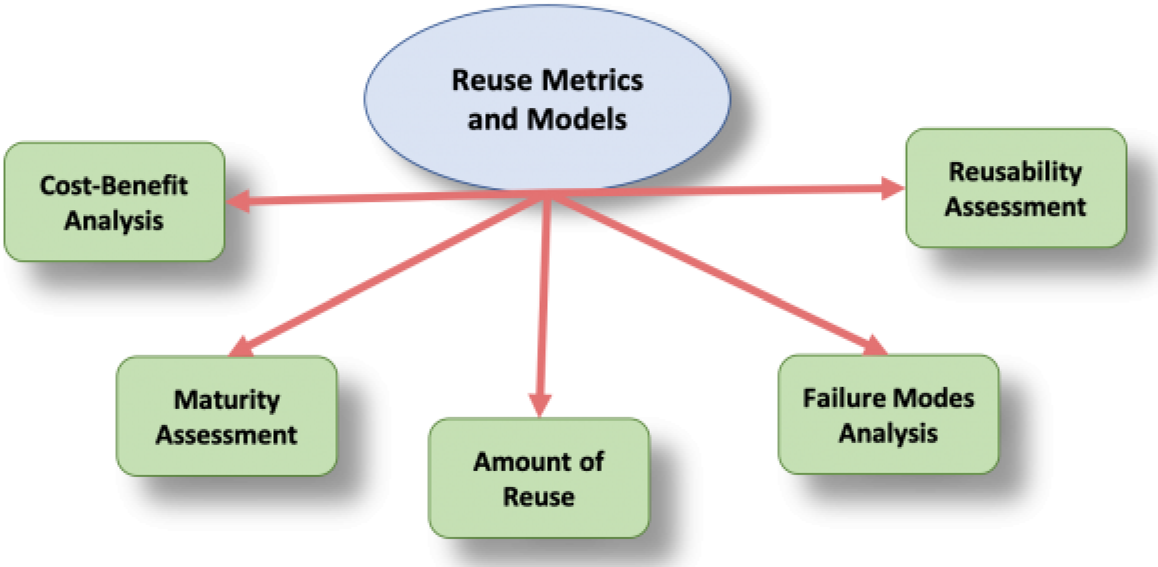


Figure 42: Categorization of Reuse Metrics and Models [3)]

Table 9: Descriptions of Reuse Metrics and Models [3)]

Metric or Model	Description
Cost-Benefit Analysis	Uses **models** to include economic cost/benefit analysis, as well as, quality and productivity payoffs. Maturity assessment models categorize reuse programs by how advanced they are in implementing systematic reuse.
Maturity Assessment	Uses **models** to categorize reuse programs by how advanced they are in implementing systematic reuse.
Amount of Reuse	Uses **metrics** to assess and monitor a reuse improvement effort by tracking the percentages of reuse for life cycle objects.
Failure Modes Analysis	Identifies and orders the impediments to reuse in a given organization.
Reusability Metrics	Indicates the likelihood that an artifact is reusable.
Reuse Library Metrics	Manages and Tracks usage of a reuse repository.

* **Note:** Organizations often encounter the need for these metrics and models in the order presented.

DIDO Specifics

Return to Top

Startup initialization is minimal because of DDS Discovery.

To be added/expanded in future revisions of the DIDO RA

[197] Luis Zafra, Know the Difference: Reusability vs. Reuse, Global Logic, 22 September 2015, Accessed 3 August 2020, https://www.globallogic.com/latam/blog/know-the-difference-reusability-vs-reuse/

[198] Jones, C. Software return on investment preliminary analysis, 1993, Software Productivity Research, Inc.,

[199] William Frakes and Carol Terry, Software Reuse and Reusability Metrics and Models, Virginia Tech, Accessed on 4 August 2020, https://pdfs.semanticscholar.org/53d3/37f49c7d1ef98968ae5ed7e699096974db10.pdf

4.3.3.3 Analysability

Return to Maintainability

About

Analysability is defined by the IEEE glossary of Software Engineering as “the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to changes to the environment”.

One way to understand the Analysability of a system or a program is to understand the size of the system or program. As a rule of thumb, as the size of system or program increases, it becomes increasingly harder to successfully modify the software to correct faults, improve performance, or to adapt to changes in the operating environment. This Analysability can be performed any time while using either the Waterfall Model or the Agile Model during the early stages of analysis and design. As projects mature and use a larger code base, the models can be based directly on either the source code or Unified Modeling Language (UML) models created using reverse engineering. The metrics can also be used during both Greenfield or Brownfield deployments. Another way to examine Analysability could be to collect and use similar metrics on Distributed Computing systems where, instead of using classes (i.e., as in Object-Oriented Programming (OOP)), the number of Nodes, Endpoints and message types could be used. To address **Structural Complexity**, the connections between processes can be used.

Table 10: Metrics for Class complexity[200)]

Type of Metrics	Metric definition
Size Metrics	
Number of Classes (NC)	The total number of classes
Number of Attributes (NA)	The total number of attributes
Number of Methods (NM)	The total number of methods
Structural complexity Metrics	
Number of Associations (NAssoc)	The total number of associations
Number of Aggregations (NAgg)	The total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship)
Number of Dependencies (NDep)	The total number of dependency relationships
Number of Generalisations (NGen)	The total number of generalization relationships within a class diagram (each parent-child pair in a generalization relationship)
Number of Generalization hierarchies (NGenH)	The total number of generalization hierarchies in a class diagram
Maximum DIT (MaxDIT)	It is the maximum DIT value obtained for each class of the class diagram. The DIT value for a class within a generalization hierarchy is the longest path from the class to the root of the hierarchy

Type of Metrics	Metric definition
Size Metrics	
Maximum HAgg (MaxHAgg)	It is the maximum HAgg value obtained for each class of the class diagram. The HAgg value for a class within an aggregation hierarchy is the longest path from the class to the leaves

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[200] Marcela Genero, Mario Piattini and Ronda de Calatrava, Empirical validation of measures for class diagram structural complexity through controlled experiments, Accessed 4 August 2020, https://pdfs.semanticscholar.org/dd52/5d80c1f370258e56cd956bcb903706c216dc.pdf

4.3.3.4 Modifiability

Return to Maintainability

About

Return to Top

Modifiability is characteristic of a system to successfully support:

- Extensibility
- Updateability
- Deletability

In other words, it is a system or products ability and receptiveness to adapt to future and often unexpected changes. Planning for Modifiability is a bit like looking into a crystal ball about the product, its place within the ecosystem and its ability to adapt to changes in the environment. An easy way to consider Modifiability of a system or product is to ask the following series of questions suggested by McGovern et al. [201]:

Table 11: Questions to consider when assessing Modifiability.

Questions proposed by McGovern et al.	**Considerations**
How often is it expected that a system change will be required?	This question is trying to understand the maturity (see 4.3.2.1 Maturity) associated with the system or product. It is not just about the product, but the maturity of the domain knowledge surrounding the system or product. For example, the accounting domain has been around for thousands of years and is well documented and understood while the use of applications to address Blockchains is less than a decade old.
What is the usual extent of the change?	This question also relates to maturity (see 4.3.2.1 Maturity) of the domain associated with the system or product. It also has to do with how conservative the attitude is towards changes. For example, changes made to an end-user entertainment application such as TikTok are easily tolerated while changes made to accounting records, which can adversely effect an individual's wealth, are frowned upon.

Questions proposed by McGovern et al.	Considerations
Who is expected to make the changes?	If changes are made by a single individual with minimal review and testing, the ability to modify the system is high, but the risk of failure is increased. Modifiability must consider these risks in the context of the domain. In essence, the better the governance over changes, the higher the probability of success (see 3 Governance). An individual can be extremely disciplined and positive when it comes to adapting to changes, while organizations can be sloppy. So, the maturity of the domain is important and the organization (even if it's a single person) is important. See ISO 90003-2018 and Capability Maturity Model Integration (CMMI).
Is it necessary for the system to use current platform versions?	This addresses the End-of-life (EoL) issues associated with any product (see 4.3.5 Manageability and 4.3.1.3 Replaceability). As the system ages, more and more EoL problems arise. As more Commercial Off-The-Shelf (COTS), Government Off-The-Shelf (GOTS), Modified Off-The-Shelf (MOTS), and NATO Off-The-Shelf (NOTS) products are used by the system and the longer the system exists the risk to the system increases because each subsystem, component or modular needs to be managed. As a case in point, in mid-2020, roughly 200 million PCs worldwide will still be running older Windows versions, mostly Windows 7[https://www.zdnet.com/article/how-many-pcs-are-still-running-windows-7-today/]]. Many of these are probably not modifiable any more.

Zarnekow et al. [202)] did a detailed study of 30 applications in 2015 and found the following time and cost characteristics and that over half (55%) of the cost of the projects can be attributed to maintenance and support. These findings underline the importance of Modifiability. All too often when a project gets started, too many problems are "kicked down the road" with the idea that "we'll cross the bridge when we get there".

Table 12: Summary of time and cost characteristics found in 30 projects (Zarnekow et al.)

			Time			Actual Cost (in Mill of Euro)					
	# of Users	# of Transactions/yr	Total	Init Dev	Prod	Total Cost	Planning	Init Dev	More Dev	Prod	Shutdown
Minimum	160	23,900	2.0	0.3	0.4	0.50	0.01	0.13	0.00	0.14	0.00
Maximum	135,500	91,250,000,000	16.4	3.0	12.4	137.33	50.00	85.00	38.00	117.8	0.13
Average			5.6	1.7	3.1	35.74	2.97	11.57	5.52	15.6	0.08
% of total			100.0%	30.3%	55.3%	100.00%	8.31%	32.37%	15.44%	43.65%	0.2%

Another paper published by Björklund [203)] reported the cost of software maintenance as 67%.

Table 13: The Cost of Software Maintenance for one project

Lifecycle Phase	Percent of Cos
Requirements Definition	3%
Preliminary Design	3%
Detailed Design	5%
Implementation	7%
Testing	15%

Lifecycle Phase	Percent of Cos
Maintenance	67%

* **Note:**

- Another study found at least 50% of the effort spent on maintenance
- Another study found between 65% and 75% on maintenance
- In embedded real-time systems, maintenance costs may be up to 4 times development costs

Regardless of the actual numbers for a system or a program, all the numbers point to one conclusion: the cost of maintenance is a major driver in the total cost of ownership for systems or projects. Much of the maintenance cost for many projects can be traced back to not planning or considering Modifiability throughout the project lifecycle, especially early on. Modifiability is closely correlated to creating layered, modular and loosely coupled systems or programs. Fortunately, there are tools which can analyze a system or a program during all its phases (see 4.3.3.1 Modularity and 4.3.5 Manageability).

Layering involves separating the system or programs based on technical responsibilities, usually using an N-Tier Architecture. Generally, these tiers are referred to as the *presentation tier*, *processing tier* and *data management tier*. Often the tiers are both logically and physically separated, with each tier running on its own dedicated platforms. In a distributed system, the tiers do not follow the client-server architecture but use a Peer to Peer (P2P) architecture. However, the peers can be categorized as fulfilling presentation, processing, and data management functionality.

In addition, systems or programs that are declarative and configurable are more modifiable, especially if the configuration describes the details of connectivity between the modules (or peers). In other words, these descriptions provide the context and should address the 5-Ws of **who**, **what**, **when**, **where** and **why**; as well as, the **how**. For example:

Table 14: How the 5-Ws and H can be used to help ensure Modifiability

W	context
who	Who can access the module (peer) including privileges: developer, a business user, an analyst, or some combination of these is responsible
what	What is the module (peer) name, version, download URI
when	When can the module (peer) be accessed: event, calendar, time, etc.
where	Where can the module (peer) be found: paths, endpoints, etc.
why	Why is the module (peer) defined: documentation, rules, filters, etc.
how	How is the module (peer) accessed: Library, RESTFul, Remote Procedure Call (RPC), DDS, Message queue, etc.

Expectations of frequent changes driven by business-related changes can be more modifiable if the rules are not codified into software but stored as machine readable rules that can be interpreted at run time using, for example, rule engines. However, the downside of a data-rule driven system is that changes in data or rules can lead to crashes and adverse impacts to stability (see 4.3.5 Manageability).

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[201] James McGovern, Sameer Tyagi, Michael E. Stevens and Sunil Mathew, Java Web Services Architecture, 2003, ISBN 978-1-55860-900-6, Accessed 5 August 2020 https://www.sciencedirect.com/book/9781558609006/java-web-services-architecture

[202] Ruediger Zarnekow and Walter Brenner, Distribution of Cost Over the Application Lifecycle - A Multi-Case Study, University of St. Gallen, 22 July 2015, Accessed 5 August 2020, Researchgate

[203] Carl Björklund, App Maintenance Cost Can Be Three Times Higher than Development Cost, 15 April 2019, Accessed 5 August 2020, https://www.econnectivity.se/app-maintenance-cost-can-be-three-times-higher-than-development-cost/

4.3.3.5 Testability

Return to Maintainability

About

Return to Top

Testability, *Testable*, *Testing* and *Test* are not synonyms for each other. Just because a system or program is undergoing *testing* using various *tests* does not necessarily mean that the system or program is actually *Testable*. The following table provides definitions for each of these four terms, associates each with the appropriate Structured Assurance Case, as well as, level in the Cognitive Model's Science and Knowledge Management DIKW (Data, Information, Knowledge and Wisdom) pyramid.

DIKW pyramid level	Structured Assurance Case	Term	Description

DIKW pyramid level	Structured Assurance Case	Term	Description
Understanding	Software Assurance (SwA)	Testability	**Testability** is about documenting the functionality and requirements for a system or program and verifying that these requirements will be or have been met. Functional requirements are generally not a problem since most of these are directly measurable or observable. Functional requirements are often directly measured or observed: a data field needing to be provided, a relationship existing between two pieces of data, or a relationship that is one-to-many or a many-to-many. For example, every person must have a unique company ID number but they may have multiple phone numbers and also belong to multiple organizations. Other functional requirements are not so definite, but expressed in terms of a range of acceptable values. For example, a Graphical User Interface (GUI) will respond in less than 5 seconds or the heart pulse rate is between 35 to 200 beats per minute. In contrast, non-functional requirements are generally more abstract: they relate to the quality of the system or program being delivered (i.e., portable, reliable, maintainable, securable, scalable, etc.) and are usually not directly measurable or observable but must be inferred from characteristics found in the delivered system's or product's architecture, design and implementation. These kinds of requirements require ways to characterize assurance and are specified in terms of claims (i.e., the system has a High Availability), sub-claims, and arguments (i.e., Availability can be predicted using a Mean Time To Repair (MTTR) of 5 minutes, 15 seconds or less of downtime in a year for all components). These kinds of requirements are generally specified in Performance or Functional Specifications. These specifications tend to focus on hardware specifications; however, performance specifications can also capture non-functional metrics. • **Note** Testability metrics are not limited to operational systems or programs but can also take advantage of system or program level artifacts that describe architecture, design, discussion papers, outside references, software and executables. Here is a list of some common “mistakes” found in requirement documents[23] that can make it difficult to determine if requirements are actually “testable”: • **Noise:** Text containing no information relevant to any aspect of the problem. For example, a requirement on a standalone application that does not need access to the Ethernet ∘ The system shall conform to IPV6 … • **Silence:** A feature not covered by any text within the Requirements documents or specifications • **Over-specification:** Description of the solution rather than the problem. For example, ∘ The distributed system must use blockchain. (blockchain is one of many distributed technologies used by Cryptocurrencies) ∘ The system must use a checkbox to select the appropriate option • **Contradictory:** Mutually incompatible descriptions of the same feature. For example, ∘ The system shall not record any personal information ∘ The system shall record all transactions and parties participating in the transaction • **Ambiguity:** Text that can be interpreted more than one way ∘ The system shall support real-time operations (what is real-time?) • **Forward reference:** Referring to a feature not yet described ∘ The system shall publish all information on a topic (but topic has not been officially defined yet) • **Wishful thinking:** Defining a feature that can’t be validated ∘ The system shall initialize all values with intelligent default choices. (what's the metric for “intelligent”?) • **Weak phrases:** Causing uncertainty (“adequate”, “usually”, “etc.”) For example, ∘ *When possible*, the systems shall … ∘ The system shall collect their data (whose data?) • **Jigsaw puzzles:** Requirements distributed across a document and then cross-referenced • **Duckspeak:** Requirements included merely to conform to standards that have no or little relationship to the problem at hand. Perhaps required as part of a boilerplate. • **Terminology invention:** “user input/presentation function”; “airplane reservation data validation function”. For example, ∘ The system shall use a double blind logged journal entry (huh, what is that?) • **Putting the onus on developers and testers:** to guess what the requirements really are. ∘ The system shall use a right-handed approach when presenting data

DIKW pyramid level	Structured Assurance Case	Term	Description
Knowledge	Claim	Testable	A **Testable** attribute of a system or program is a functional or nonfunctional requirement that may be testable or not. Some requirements can be directly tested for by running specific tests (i.e., Unit Testing, integration testing, etc.) using test plans that exercise a portion of the system or program software responsible for providing specific functionality. For example, the system is supposed to offer the choice of none, one, and many. Another example might be that when an option is selected, a message is sent out over the network. By design, some requirements are not directly testable, i.e, are untestable. Often, these requirements are met through the use of mathematical proofs or demonstrations. For example, the generation of a Universally Unique IDentifier (UUID) can not be tested directly; instead, the algorithm used to generate them must provide an explanation and proof that no two sets of conditions will produce the same UUID. Often there is a risk of generating the same UUID, but the chances of the same UUID being used in identical domains or environments is even smaller. Another example would be the reCAPTCHA, which shows a series of photos and asks the user to identify the ones with green peas in them. The order of the photos and the thing it is asking you to identify are randomly assigned.
Information	Argument	Testing	Testing is a process that generally involves the execution of the system or program under scripted, controlled situations. The scripts can be human instructions in documents or they can be captured in text files that a testing engine uses to drive the software. Sometimes, a Unit Test is used to test individual modules before they are integrated into the system or program. Below are the various requirement conformity checks that can be performed to verify functional requirements: 1. Unit Testing 2. Integration Testing 3. End-to-End Testing (E2E Testing) 4. Smoke Testing 5. Sanity Testing 6. Regression Testing 7. Acceptance Testing 8. White Box Testing 9. Black Box Testing 10. Interface Testing 11. Interoperability Testing
Data	Evidence	Test	*Test* refers to the act of collecting the evidence used to support arguments, sub-claims and claims made about the system or program. There is not a one-to-one relationship between a Test, an Argument, a Sub-Claim or a Claim. Instead, one piece of data can support multiple Arguments and an Argument can support multiple Sub-Claims or Claims. That is why it is so important to have a Structured Assurance Case Model.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[23] Achieving Requirements Testability, ProlificsTesting, 10 October 2018 Accessed on 9 August 2020 https://www.prolifics-testing.com/news/achieving-requirements-testability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:20_maintainability:testability

Last update: **2021/06/16 11:46**

4.3.4 Securability

Return to Non-Functional Requirements

About

Security is not a single “thing” that can be added to a system. To be truly secure, the entire End-to-End Solution (E2ES) needs to be secure and needs to be considered during the entire System Lifecycle. As shown in Figure 43, a layered approach is used to help isolate the security needs. Each layer represents a portion of the Information Technology (IT) stack, including the people who use and have access to the IT stack.

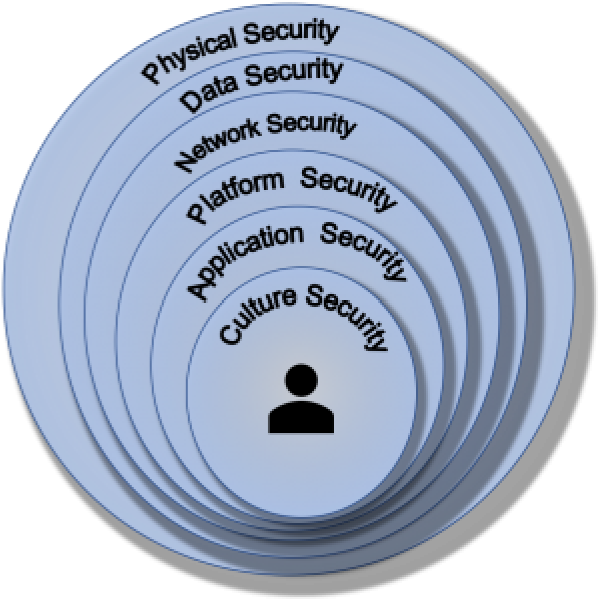


Figure 43: The layers of security.

Table 15: Definitions for layers of security

Layer	Description
Physical Security	The physical security is concerned with preventing physical harm to the Computing Platform (e.g., theft, fire, flooding, etc.), as well as, preventing access to the physical platform via “back doors” thereby allowing breaches by potentially malicious actors (e.g., using pluggable USB drives, adding wire sniffers to the network, or the internal threat posed by employees with access).
Data Security	Data security ensures that Data at Rest, Data in Motion, or Data in Use remains intact (i.e., completeness, accuracy and consistency). For example, allowing incomplete data to be stored (i.e., date of a transaction, or *authorized by* fields). Roundoff Error can also affect the accuracy of the data. Modifying a bank account balance introduces inconsistencies that can be detected.

Layer	Description
Network Security	Network security issues are generally the result of unaddressed network vulnerabilities. There are three main network categories for vulnerabilities: software, hardware, or organizational processes. Software and hardware that are not kept current are subject to malicious attacks by merely exploiting known vulnerabilities. Another issue for networks are social engineering attacks where people violate hardware and software protection policy and procedures to compromise the data. The first line of defense for networks is the use of a Hardware Firewall, which predominately protects the Network Nodes inside a Local Area Network (LAN) from external Nodes on the Internet. Another common tool is the use of Networking Access Control Lists (ACLs).
Platform Security	Platform security involves an attack on a Computing Platform by the introduction of malicious software, the modification of access controls or configuration data, or having incorrectly configured settings (i.e., Software Firewall, Access Control List (ACL), Full-Disk Encryption (FDE)). Many platforms can require authorization of peripheral functions such as geo-location services, Bluetooth, TCP/IP networks, radio networks, and segmented portions of the disk storage such as photos.
Application Security	Application Security features include authentication of users using multi-factor authentication such as user id and passwords, asking additional questions only the user knows the answer to, use of a One-Time PIN (OTP) to a known device, a fingerprint or face recognition. Application Security also includes authorization that maps the user's identity with a list of applications the user can access and even the privileges the user has within the application (read/write/delete, etc). Sometimes an application can encrypt information as it moves between the components of the system (CPU, Memory, Disk, network, etc.). Another key aspect is the secure logging of activities occurring within an application (e.g., user granted access, user deletes information, user updates information, etc.)
Culture Security	One of the biggest threats to any system or program is the internal threat caused by inadvertent or overt actions taken by the people within the organization. To overcome these threats, there needs to be Cultural Security (also known as cybersecurity) that seeks to change the the mindset of the authorized users. Some basic examples are not using `password` as a password, not writing the passwords on paper, having a separate password for every person, changing the password every so many days, keeping ACLs up to date and reflecting the current roles and responsibilities of the users. Sometimes, it comes down to just observing the behavior of others[205]

ISO/IEC 25010 defines Security as the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. This characteristic is composed of the following sub-characteristics[206]:

- 4.3.4.1 Confidentiality
- 4.3.4.2 Data Integrity
- 4.3.4.3 Non-Repudiation
- 4.3.4.4 Authenticity
- 4.3.4.5 Accountability

See DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[205] Cyber Security Culture in organizations, European Union Agency for Network and Information Security (ENISA), November 2017, Accessed on 13 August 2020, https://www.enisa.europa.eu/publications/cyber-security-culture-in-organisations

[206] ISO25000 Software and Data Quality ISO/IEC 25010, 2011, Accessed 13 August 2020, https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

4.3.4.1 Confidentiality

Return to Securability

About

Confidentiality is usually covered by the use of a Confidentiality Agreement or Non-Disclosure Agreement (NDA), which defines a set of rules or a promise limiting access or places restrictions on certain types of information. Areas that have legal agreements covering confidentiality are:

- Legal Confidentiality
- Medical Confidentiality
- Clinical and Counseling Psychology
- Commercial Confidentiality
- Banking Confidentiality
- Public Policy Concerns
- Religious Confidentiality

As a rule of thumb, it is best to treat all Personal Identifiable Information (PII) as confidential and to secure it (i.e., require authentication both to access the data and log access to the data).

The US National Institute of Standards and Technology (NIST) describe the kinds of data that should be treated as PII[207)] as:

- Name, such as full name, maiden name, mother‘s maiden name, or alias
- Personal identification number, such as:
  - Social security number (SSN),
  - Passport number,
  - Driver‘s license number,
  - Taxpayer identification number,
  - Patient identification number,
  - Financial account number, and
  - Credit card number

NIST also identifies information which potentially can be used to identify people:

- Address information, such as street address or email address
- Asset information, such as Internet Protocol (IP) or Media Access Control (MAC) address or other host-specific persistent static identifier that consistently links to a particular person or small, well-defined group of people
- Telephone numbers, including mobile, business, and personal numbers
- Personal characteristics, including photographic image (especially of face or other distinguishing characteristic), x-rays, fingerprints, or other Biometric image or template data (e.g., retina scan, voice signature, facial geometry)
- Information identifying personally owned property, such as vehicle registration number or title

number and related information

- Information about an individual that is linked or linkable to one of the above (e.g., date of birth, place of birth, race, religion, weight, activities, geographical indicators, employment information, medical information, education information, financial information).

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[207] Erika McCallister Tim Grance and Karen Scarfone, Guide to Protecting the Confidentiality of Personally Identifiable Information (PII), Special Publication 800-122, April 2010, Accessed on 13 August 2020, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-122.pdf

4.3.4.2 Data Integrity

Return to Securability

About

Return to Top

Data Integrity is the completeness, accuracy and consistency of data throughout the entire data lifecycle of the data as well as when the Data is at Rest, Data in Motion and Data in Use.[208)]

Figure 44 shows the five levels Automation Pyramid and the functionality usually associated with each one. There is Data at Rest at each level of the pyramid. As the data transitions up and down from level to level within the pyramid, the Data is in Motion. Within each level, the data will most likely be accessed therefore, the Data is in Use.

Table 16: The five levels of the Automation Pyramid.

Automation Level	Description
Field Level	The Field Level where products are produced. In other words, this is where the physical work plus monitoring occur. Electric motors, hydraulic and pneumatic actuators to move machinery, proximity switches used to detect that movement or certain materials, photoelectric switches that detect similar things will all play a part in the field level.
Control Level	The Control Level uses the control devices to “run” the devices in the Field Level. The Control Devices make decisions based on information provided by sensors, switches, and other input devices to complete the programmed task.
Supervisory Level	The Supervisory Control and Data Acquisition (SCADA) is combines the Field and Control Levels to provide oversight from a single location. This is usually accomplished using Graphical User Interface (GUI), or Human-machine interface (HMI), to remotely control operations. For example, water plants often employ this technology to control remote water pumps.
Planning Level	The Planning Level uses Manufacturing Execution System (MES) to monitor the entire manufacturing process. For example, in a factory to plan for everything from raw materials to the finished products. This allows management to visualize the current state of operations and aids them in making decisions and adjust raw material orders or shipment plans based on real data received from Supervisory, Control and Field Levels.
Management Level	The management level uses the companies integrated management system such as as Enterprise Resource Planning (ERP). Corporate management visualize and control operations. This level allows the businesses monitor all levels (i.e., manufacturing, to sales, to purchasing, to finance and payroll). The integration of an ERP promotes efficiency and transparency within a company by helping to communicate the levels.

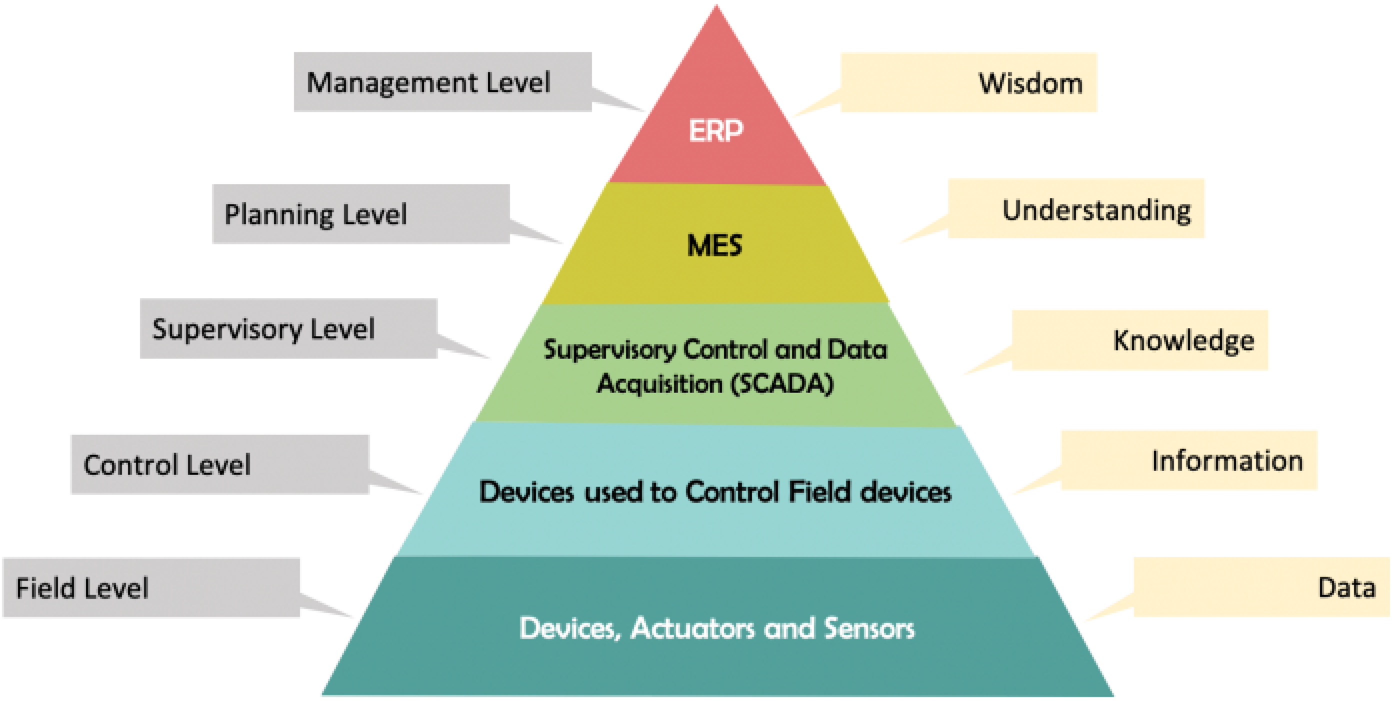


Figure 44: Automation Pyramid

At each level, the Data at Rest can be categorized as one of two kinds of data integrity both of which are a collection of processes and methods intended to enforce Data Integrity.

- Physical Integrity protects data's wholeness and accuracy as it's being used. When expected or unexpected down times occur (i.e., natural disasters strike, power goes out, or hackers disrupt database functions) physical integrity is compromised. Some other issues which can compromise the integrity of the data are Human error, storage erosion, or a host of other issues making it impossible for data processing managers, system programmers, applications programmers, and internal auditors to obtain accurate data.
- Logical Integrity keeps data unchanged as it is accessed. Logical integrity protects data from some of the same issues as Physical Integrity (i.e., human error and hackers as well) but in different ways. There are four types of logical integrity.
  1 - Entity Integrity - supports unique values that identify any particular data entry and that the key is not null.
  2 - Referential Integrity - ensures that references to other data entries exist.
  3 - Data Integrity - ensures that domain rules (i.e., data restrictions) are enforced for the data within the Data Structure. For example, minimum, maximum, number of decimals, nullable, etc. are enforced.
  4 - User Defined Integrity - ensures that business rules are enforced. For example, if a value is set, another value must also be set (unset); if a value exceeds a threshold, a notice must be sent.

Note: Data integrity is not Data Security and not Data Quality.

- Data Security defines the steps taken to prevent corruption from within and from outside attacks by people or processes.
- Data Integrity defines the steps taken to keep the data intact and accurate from internal people and processes and for the entirety of the data's existence.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA (look at spec if DDSF)

[208] What is Data Integrity, Accessed 8 July 2020, https://www.talend.com/resources/what-is-data-integrity/

4.3.4.3 Non-Repudiation

Return to Securability

About

Non-Repudiation [209] means that it is not possible to repudiate (i.e., deny) that an action has been taken. For example, the signed contract witnessed by two people could not be repudiated. In other words, the contract now has Non-Repudiation.

Non-Repudiation is about providing assurance using evidence that an action has been done. For example, a data sender is provided evidence (i.e., proof) of delivery while the receiver is provided evidence (i.e., proof) of the sender's identity. As a consequence, neither the sender or the receiver can deny having processed the data.

Non-Repudiation applies to more than just sending data between two parties. It can be applied to any action or activity. For example, by digitally signing an email, the receiver has evidence (i.e., proof) that the email is from the entity that signed the email. In other words, it is not possible to repudiate that the email came from the entity that digitally signed the email. Another example is the use of identities in configuration management systems. The change (i.e., transformation) was recorded in a log along with the identity of the individual that made the change. In this way, all changes made to the configuration have Non-Repudiation.[210]

There is a lot of overlap in Non-Repudiation and Access Control. During access to a controlled resource, the identity of the entity trying to access the resource is verified against an Access Control List (ACL). When access is allowed or denied, an entry is made into a log. Once the entry is made, the access has Non-Repudiation. In other words, once an Access Control Function is executed, there is generally sufficient evidence to for Non-Repudiation of access to the controlled resource.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[209] Non-Repudiation, Computer Security Resource Center (CSRC) Accessed 14 August 2020, https://csrc.nist.gov/glossary/term/non_repudiation

[210] Evan Wheeler, Security Risk Management, 2011, Accessed 14 August 2020, https://www.sciencedirect.com/science/article/pii/B9781597496155000074

4.3.4.4 Authenticity

Return to Securability

About

Authenticity is a property indicating the source and origin of the information[211]. The process of authenticating a source starts when an entity (i.e., user, remote process, intelligent agent, etc.) attempts to access resources on a Computing Platform. The entity proves their identity in order to gain access rights. For example, traditionally when logging into a computer, users use a Single-Factor Authentication (SFA) by providing a `usernames` and `passwords` to confirm their identity to allow future authentication for access to resources. However, this `usernames` and `passwords` login combination is no longer considered secure enough, especially if there are poor CyberSecurity Culture (CSC). As a consequence, many systems have added Two-Factor Authentication (2FA) that require Biometrics (i.e., facial recognition, fingerprints, etc) or One-Time PIN (OTP). These 2FA methods generally require the user to be physically present to successfully login.

Public Key Infrastructure (PKI) is generally used to connect servers and clients or even nodes that have no user present to perform the SFA or the 2FA methods of authentication. It is often incorrectly used as a synonym for Encryption. Encryption is an algorithm used to encrypt and decrypt data. PKI is an infrastructure built around asymmetric encryption with two keys: public and private. PKI is used extensively to securely transfer data between Network Nodes. In the PKI infrastructure, entities (i.e., AAA and BBB) exchange public keys. To exchange information, one entity (i.e., AAA) encrypt a document using the other entities (i.e., BBB) public key. Anyone can receive the document encrypted by AAA using BBB's public key, but it remains encrypted until BBB uses the private key in the PKI to decrypt the document.

PKI is the backbone of most of the major secure document exchange sites. Some examples are[212]:

- Securing emails - Email Security (S/MIME Protocol)
- Securing web communications - Website Security
  - Hypertext Transport Protocol Secure (HTTPS)
  - Secure Sockets Layer (SSL)
  - Transport layer security (TLS)
- Secure Shell Protocol (SSH)
- Digitally signing software, applications or data
- Encrypting and decrypting data
- Smart Card authentication
- Subscriber Identity Module (SIM)

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

=-

[211] Authenticity, Computer Security Resource Center (CSRC) Accessed 14 August 2020, https://csrc.nist.gov/glossary/term/authenticity

[212] How Does PKI Work ?, Venafi, Accessed 14 August 2020, https://www.venafi.com/education-center/pki/how-does-pki-work

4.3.4.5 Accountability

Return to Securability

About

Accountability is the principle holding an individual entrusted to safeguard and control key components of a system or program (i.e., equipment, keying material, and information) answerable to proper authority for the loss or misuse of that component.[213)]

Accountability is a security goal outlined in ISO/IEC 24010[214)] requiring the actions of an entity to be traced uniquely to that entity. Accountability directly supports Non-Repudiation. It also provides a deterrence, helps with fault isolation, and is useful in intrusion detection and prevention. In many cases, it is a key source of evidence use in and After Action Review (AAR) and can ultimately, if needed, support legal actions.

Accountability is part of an information security plan. The plan should enumerate every individual working with an information system and define specific responsibilities (i.e., tasks) regarding information assurance. Each task needs to me measurable and be subject to oversight by individuals higher up in the command chain.

One example, might be an information security requirement that holds all employees responsible for not installing software from any source other than a company-owned repository (i.e., a task). The individual responsible for upholding the requirement might perform a periodic check of corporate assets to determine that the policy is being followed. Any violations in the requirement would hold the individual responsible for performing the task. The plan makes the individuals aware of the tasks expected of them, and guide continual improvement in compliance with the requirement.[215)]

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[213)] Accountability, Computer Security Resource Center (CSRC) Accessed 14 August 2020, https://csrc.nist.gov/glossary/term/accountability

[214)] Accessed 15 August 2020, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&start=6

[215)] Computer-Security-Glossary.org, Accessed 15 August 2020, https://www.computer-security-glossary.org/accountability.html

4.3.5 Manageability

Return to Non-Functional Requirements

Manageability is most important during the second half of System Lifecycle phases (i.e. operation, maintenance, support). Manageability can greatly influence the recurring costs and can increase the chances of a failure. Often a system that is hard to manage is described as fragile since the smallest change can have dire consequences on the system's functionality.

> *Manageability directly influences a system's reliability, availability, security, and safety; therefore, it is a key ingredient of system dependability.*
>
> *Just like security and safety, manageability is generally hard to retrofit in complex systems—it is always easier to build it in from day one. However, in the absence of means to measure manageability and quantify the various tradeoffs, it is difficult to get the design right. We proposed a manageability metric that combines management workloads and weightings based on real world studies with direct measurement of the number of steps involved in management tasks and their duration.* [216]

- 4.3.5.1 Types of Manageability Functions
- 4.3.5.2 Manageability Costs
- 4.3.5.3 System Manageability Issues
- 4.3.5.4 Software Manageability Issues

DIDO Specifics

Return to the Top

> To be added/expanded in future revisions of the DIDO RA

[216] Toward Quantifying System Manageability, George Cadea, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, Accessed 20 July 2020, https://www.usenix.org/legacy/event/hotdep08/tech/full_papers/candea/candea_html/index.html

4.3.5.1 Types of Manageability Functions

Return to the Manageability

About

Return to Top

NI (formerly National Instruments) defines four basic manageability functions [217] described in the following Table:

Table 17: The kinds of Management Functions needed for Manageability

Kinds of Management	**Description**
Health Monitoring, Logging, and Alerting	During operations a key part of managing the system is the collection of metrics about the operations that indicate the health of the system. • Monitoring can include usage of Central Processing Unit (CPU), memory, energy, cache, heat, on-line and off-line storage, network connectivity, network loads of particular computers. In a distributed system, this also includes the metrics associated with the connectivity of the nodes that participate in the distributed system (i.e., the latency, lag, error rates, upload and download speeds, etc). • Information about the overall operations need to be handled. For example, data used to Trace, and Debug; provide Informational context; and alert to warnings, errors and fatal situations.
Configuration and Control	During the startup and sometimes during operations, the system needs to obtain configuration details and provide control over the systems as a whole or to the individual components. This becomes even more important in a system that is distributed on a system of nodes. [1] • Typical configuration management data are Internet Protocol address (IP address), network ports, environment paths to files, and maximum usage limits for things like CPU and disk space. • Typical commands are start, pause, resume and stop. Sometimes abort is also used to capture detailed dump files that can be used for analysis.

Kinds of Management	Description
Deployment and Updates	During operations, - There is sometimes a need for the efficient and sometimes automatic deployment of system resources such as web servers, application servers, database management systems, backups etc. This is useful when systems are made as a collection of other systems but is critical when managing distributed systems. - In addition, this kind of management needed during first-time installation of systems (i.e., wizards) but also in deployed systems for applying patches, performance, and feature updates.
Asset Discovery and Inventory	In remote systems or especially in distributed systems, an accurate and timely discovery of the system or component needs to be made. - In truly distributed systems, this discovery needs to be dynamic and should easily adapt to new locations. In a distributed system this must also support the dynamic creation and destruction of nodes in the system network. - In complex systems deployed on a single computer there is a dependency on other components within the node. An inventory needs to be made of the components required as included in a package management description (i.e., manifest). - In a distributed system running on distributed nodes there is a need to understand the resources dedicated to the system. Having an automated inventory helps with asset management, containing cost controls, and scheduling maintenance or replacement of systems.

[1] **Note:** The following example of how configuarions can become a manageability issue is from Candea [1]- *The greatest manageability challenge is posed by stateful systems (e.g., databases, filesystems). By contrast, stateless applications (e.g., Web servers) require little configuration, can be scaled through mere replication, and are reboot-friendly. While one administrator can manage 100s to 1000s of Web servers, it takes approximately one administrator for each TB of data in a database. The number of knobs on stateful systems is overwhelming: the Oracle DBMS has 220 initialization parameters and 1,477 tables of system parameters, while its "Administrator's Guide" is 875 pages long.*

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[217] What is Manageability?, 5 May 2019, National Instruments (NI), Accessed 25 July 2020, https://www.ni.com/en-us/innovations/white-papers/13/what-is-manageability-.html

4.3.5.2 Manageability Costs

Return to the Manageability

About

> *It is no longer necessary to argue that managing enterprise computing systems is complex and time-consuming, or that the cost of managing Information Technology (IT) infrastructures far exceeds the hardware and software costs—numbers speak for themselves: IT operations account for 50%-80% of today's IT budgets, amounting to tens of billions of dollars yearly. Besides the bottom line, poor manageability also impacts reliability, availability, and security in harder-to-quantify ways. As human error becomes the dominant cause of unscheduled downtime, we desire systems that are easier, cheaper, and quicker to manage.* [1)]

The cost of managing systems is expensive and is getting more expensive. This is particularly true when using servers that are aging. The cost of support and administration of a new serer is about `$4,200`/year; by year seven, the cost will rise to about `$17,000`/year. It is logical to try and decrease these costs through refreshing the hardware [218)].

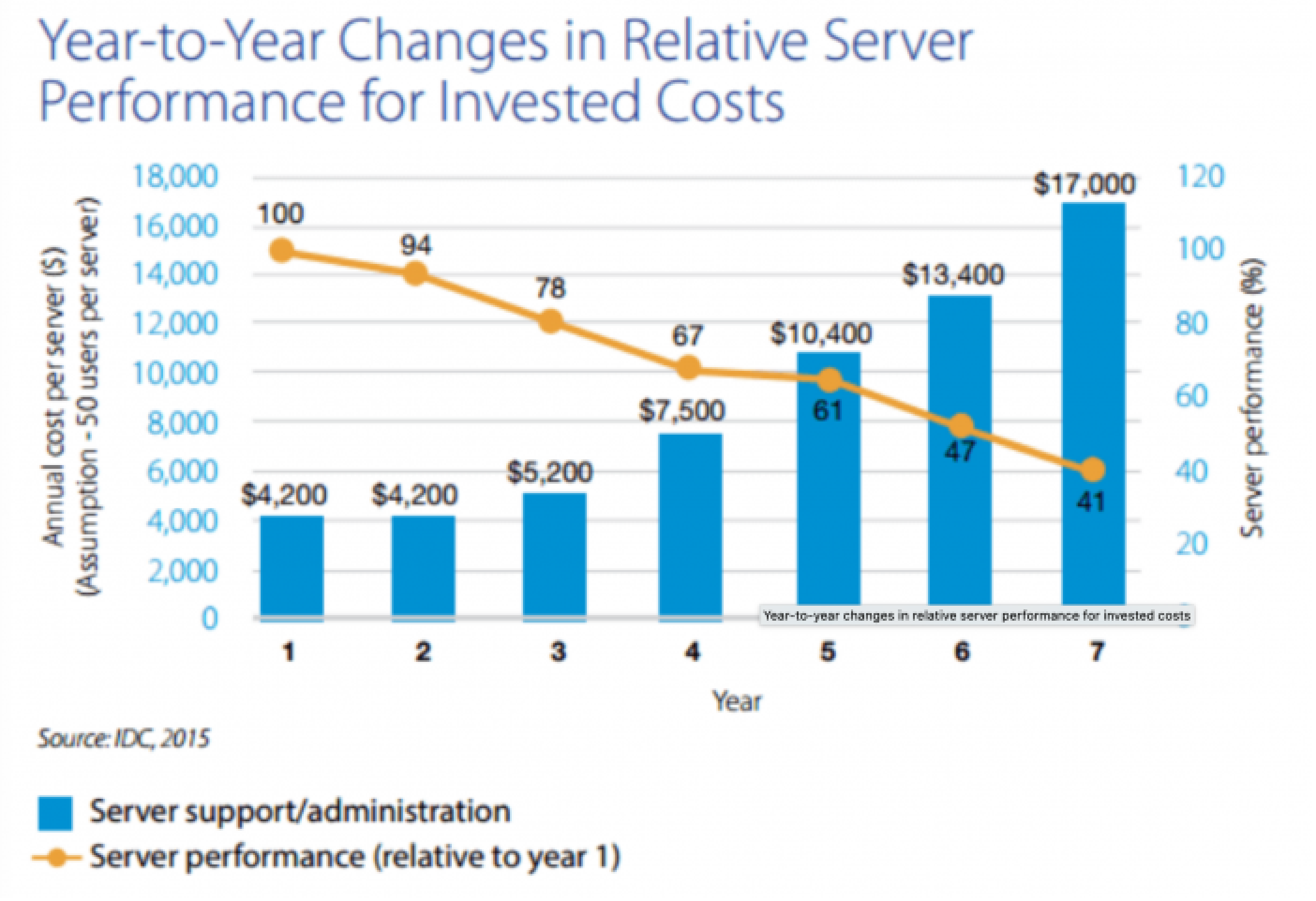


Figure 45: Year-to-Tear Changes in Relative Server Costs

Although these are the costs of owning and operating physical servers, the trend is relevant to almost all IT systems including software. The costs of managing the software is analogous to maintenance of the hardware but is probably worse since application software is dependent on many more layers of hardware and software. For example, a software system sitting on multiple Operating Systems (i.e, Linux, Unix, Windows, MacOS, IOS and Android), using a DataBase Management System (DBMS), a Web Server and using Java, Python, JavaScript, HTML, CSS, and supporting multiple browser (i.e., Firefox, Chrome and Safari) soon is overwhelmed by just keeping each system up-to-date and working. Distributed systems can compound that problem because each node in the System Network needs to be managed as well.

Granted, there are products and tools that can reduce the complexity of managing the various platforms easier and this is a big step towards helping manageability. But the overall goal is always to have systems that are easier, cheaper, and quicker to manage. There is another way to think about manageability called the Bathtub Curve. Although Figure 46 is targeted at hardware, there are similar things that effect software. Probably one of the biggest problems is that as the number of components in the system increase, the number of End-of-life (EoL) or End of Sales(EoS) issues need to be addressed. For example, a system that runs on an Operating System, has to worry about the EoL of the operating system. If it uses a database, it has a similar problem. These are generally not a problem in the early stags of the a system's lifcycle and hopefully of minimum problem during its “useful Life” phase. But as the system ages, more and more EoL problems arise. The more Commercial Off-The-Shelf (COTS), Government Off-The-Shelf (GOTS), Modified Off-The-Shelf (MOTS), and NATO Off-The-Shelf (NOTS) products used by the system, the longer the system exists there is an increase in risk to the system because each subsystem, component or modular need to be managed.

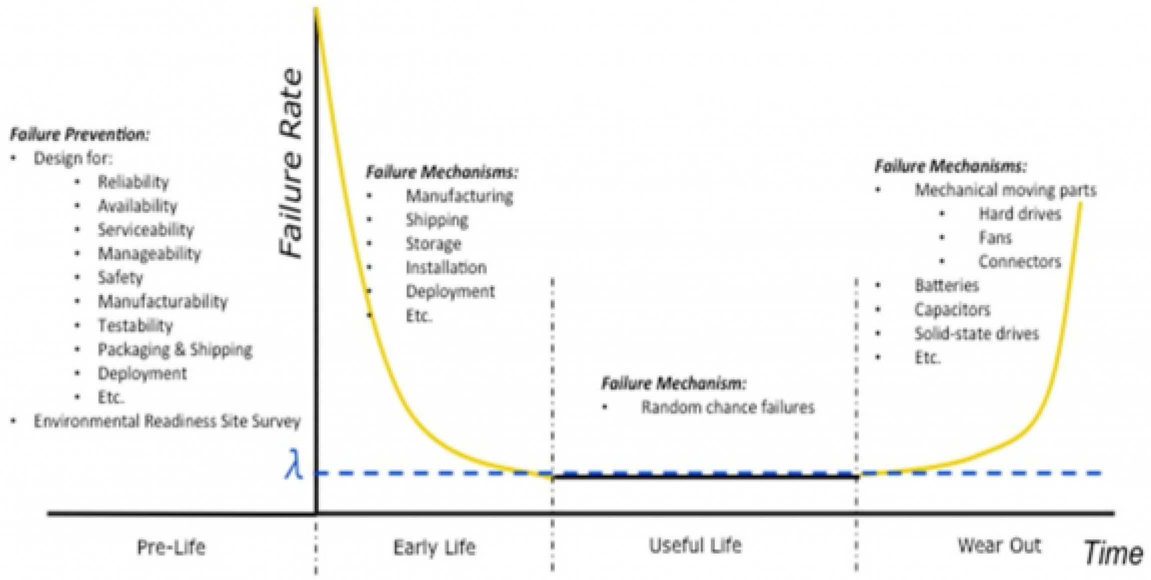


Figure 46: The Bathtub Curve [1)]

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[218] The Hidden Costs of your aging IT Infrastructure, Barry Angell, 9 January 2017, accessed 15 July 2020, https://blog.juriba.com/the-hidden-costs-of-an-aging-it-infrastructure

4.3.5.3 System Manageability Issues

Return to the Manageability

About

Subsystem, Component and Module Lifecycle Issues

"Studies have shown the average software program lifespan over the last 20 years to be around 6-8 years. Longevity increases somewhat for larger programs, so for extremely large complex programs (i.e., over a million Lines of Code – LOC) the average climbs as high as 12-14 years."[219] Obviously, there is not just the lifespan of the target system, but there are independent lifespans for each version for each subsystem, module or component that is developed externally. For example, the Windows Operating System (OS) first appeared in the mid 1980s with version 1.0.[220] and the current version is 10. About every 10 years, Microsoft release another major release of Windows. [221]. IPV4 was originally available in 1883. Windows 7 was release in October 2009 and IPV4 was the dominate Internet Protocol (IP). By 2012, IPV6 gained dominance[222]. Therefore, if your system was released in 20010 using IPV4 and Windows 7, by 2012 the network protocol needed to be upgraded which can have cascading maintenance effects throughout your system. By 2015, Windows 10 was released again having a cascading effect on upgrades.

Figure 47 developed by the Industrial Internet Consortium[223] illustrates some of the architectural components required in a generic Industrial Internet of Things (IIoT) system. When a system is deployed, each of these components needs to be managed. Each of these components has its own unique System Lifecycle which evolves independently of the target system.

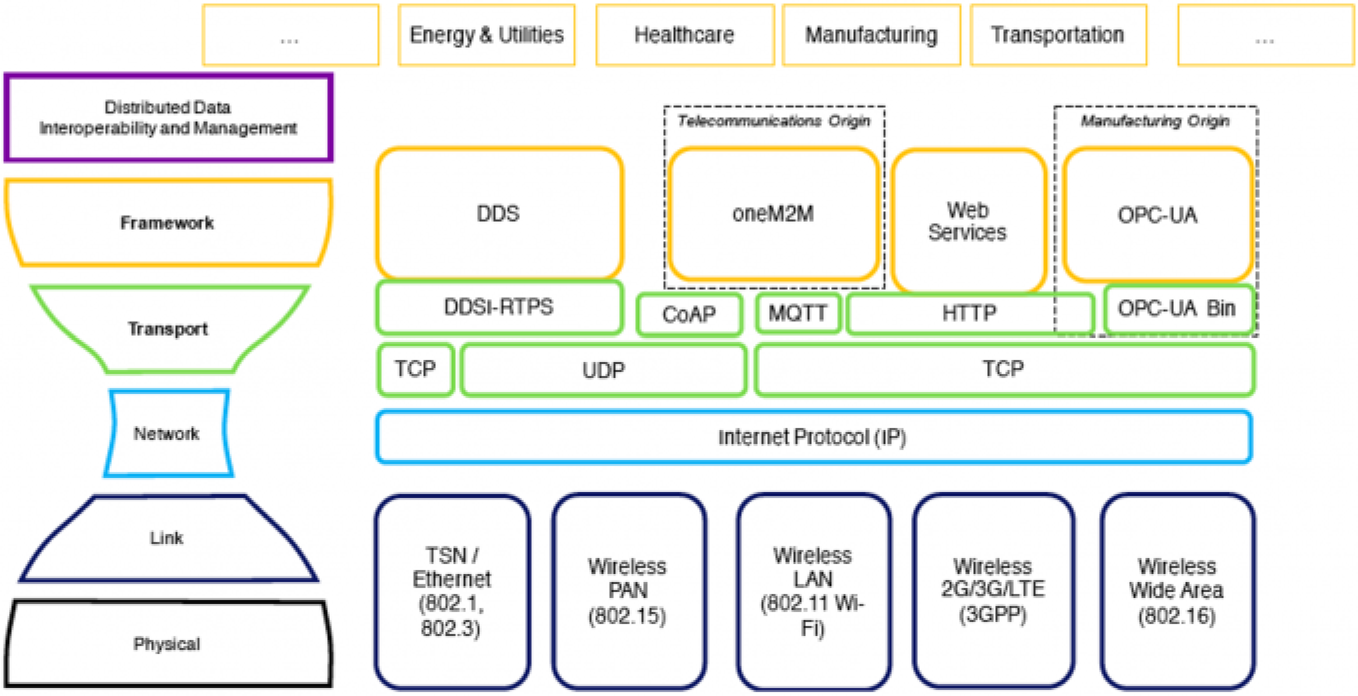


Figure 47: The Industrial Internet Consortium’s Connectivity Framework layer provides the foundation for the interoperable transfer of common structured data across systems and domains [7)]

System Monitoring

Return to the Top

The fundamental requirement for manageability of any system is the collection of data about the system. This is often done with Monitoring Software specifically designed for this task. However, the task of monitoring complex, distributed systems is often difficult and beyond the scope of any particular product. The best place to start is to think of the monitoring in terms of layers. There are considered three layers[224)].

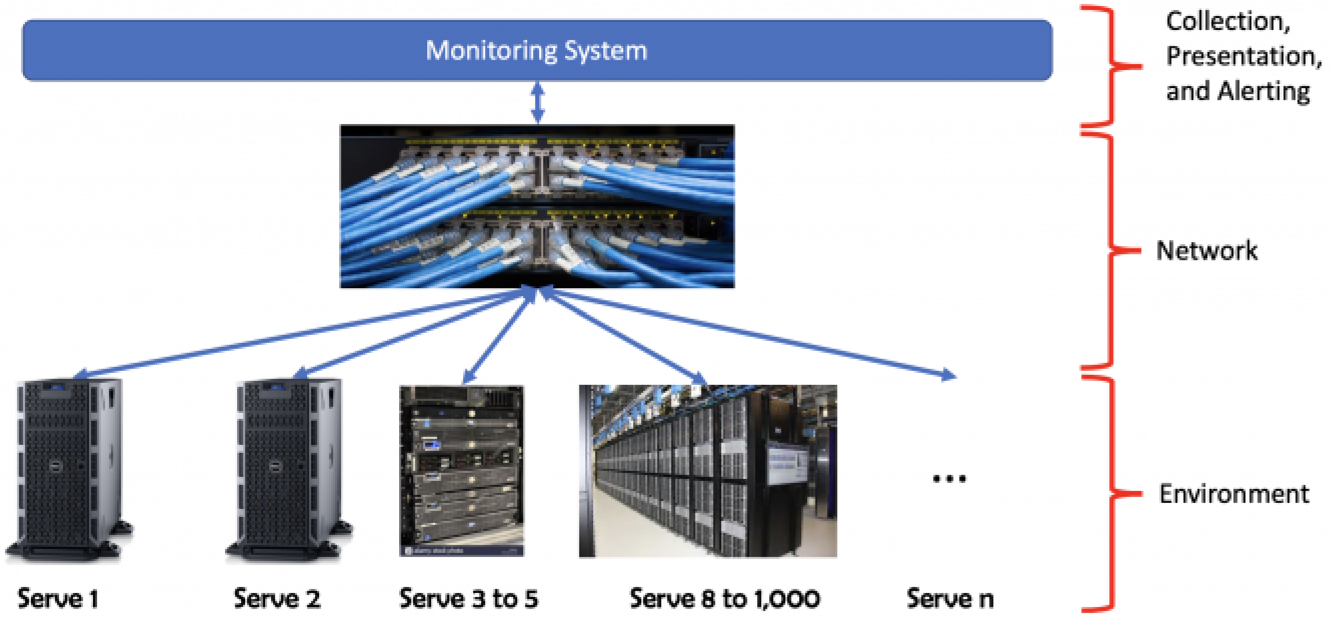


Figure 48: Three major monitoring layers representing sample design of monitoring solution

System Logging

Return to the Top

The Data Monitoring is the use of Data Logging to collect and store data for analysis to discover trends or record the events and actions of an application, a system, or a network. This allows for tracking of interactions using messages. Some of the commonly used logging levels are:

Table 18: Some common logging levels used in applications[225)]

Logging Level	Description
debug	Designates fine-grained informational events that are most useful to debug an application.
trace	Designates information about the flow of the execution or threads in an application.
info	Designates informational messages that highlight the progress of the application at coarse-grained level.
warn	Designates potentially harmful situations.
error	Designates error events that might still allow the application to continue running.
fatal	Designates very severe error events that will presumably lead the application to abort.

System Management

Return to the Top

Project Management Software is software used for project planning, scheduling, resource allocation and change management. It allows project managers (PMs), stakeholders and users to control costs and manage budgeting, quality management and documentation and also may be used as an administration system. Project management software is also used for collaboration and communication between project stakeholders. These tools can help throughout the system lifecycle, from requirement analysis through sun-setting the system. However, in a distributed system, these tools can play a significant role in determining the health of each node, the network and the overall system of nodes.

Although the publication on Software Metrics for Predicting Maintainability [226] is a bit dated, many of the ideas of capturing metrics to measure Maintainability are still relevant today. By studying these metrics and understanding the formulas and the parameters used in the formulas, a lot of insight can be provided in explaining positive and negative Manageability traits.

- **Note:** Many of these metrics were originally targeting for Systems (or projects) that used the Waterfall Model which has been replaced for many systems (or projects) with Agile Models and DevOps. Many of these metrics can and should be applied to each Sprint to make sure the qualities of the system are maintainable and manageable. In a Distributed Application (ÐApp or DApp), word module can be replaced with subsystem or Node.

Table 19: Description of Mectrics for Maintainability of Systems (or Projects) [10]

5.1	UR	Un-referenced Requirements	The number of original requirements not referenced by a lower document in the documentation hierarchy.
5.2	NR	Non-Referencing Items	The number of items not referencing an original requirement.
5.3	M-MC	Module Coupling	A measure of the strength of the relationships between modules.
5.4	M-MS	Module Strength	A measure of how strongly related are the elements within a module.
5.5	HK-IF	Information Flow	A measure of the control flow and data flow between modules.
5.6	R-IF	Integrated Information Flow of Rombach	A measure of inter-module and intra-module complexity based on information flow.
5.7	KPL-IF	Information Flow by Kitchenham et al	A measure of inter-module complexity inspired from Henry & Kafura's information flow metric. Since Kitchenham et al experienced some difficulties in understanding the definition of flows provided by Henry & Kafura, they formulated a new set of definitions.
5.8	IF4	Information Flow Complexity	A measure of inter-module complexity based on information flow
5.9	CA-DC	Design Complexity of Card & Agresti	A measure of inter-module and intra-module complexity of a system based on fan-out, number of modules and input/output variables
5.10	COCO	Cocomo Inspired Metric	A selection of appropriate adjustment factors of the intermediate Cocomo metric
5.11	v(G)	Cyclomatic Complexity Number	The number of independent basic paths in a program.
5.12	knots		The number of crossing lines (unstructured `goto` statements) in a control flow
5.13	RLC	Relative Logical Complexity	The number of binary decisions divided by the number of statements
5.14	Vcd	Comments Volume of Declarations	Total number of characters found in the comments of the declaration section of a module. The declaration section comprises comments before the module heading up to the first executable statement of the module body.

5.15	Vcs	Comments Volume of Structures	Total number of characters in the comments found anywhere in the module except in the declaration section. The declaration section comprises comments before the module heading up to the first executable statement of the module body.
5.16	Ls	Average Length of Variable Names	Mean number of characters of all variables used in a module. Unused declared variables are not included.
5.17	LOC	Lines of Code	The number of lines in the source code excluding blank lines or comment lines.
5.18	E	Software Science Effort	An estimation of programming effort based on the number of operators and operands. It is a combination of other Software Science metrics.
5.19	DAR	Documentation Accuracy Ratio	A verification of the accuracy of the CEI Spec, RS and SDD with respect to the source code.
5.20	SCC	Source Code Consistency	The extent to which the source code contains uniform notation, terminology and symbology within itself.

Vendor Lock-in Issues

Return to the Top

A major management issue for many projects is the avoidance of Vendor Lock-In. Vendor lock-in restricts the options available to a system (or project) because of the dependency on sole-source proprietary technology, solution or service provided by a single vendor or vendor partner. This technique can be disabling and demoralizing because customers are effectively prevented from switching to alternate sources for the technology, solution or service making the customer-vendor relationship one sided.

Vendor Lock-In reduces the ability of manage costs over the life expectancy of the system (or project) or avoid risks when a vendor of a product ceases to maintain a critical component of the system (or project) or even when the product ceases to exist.

This can be partially mitigated through the use of Open Source Software (OSS), however, remember, just because something is OSS, does not mean there is not a vendor. Additionally, if the OSS software is deprecated or evolves in a divergent way from the requirements of the target system (or project), then the responsibility for the care and maintenance of the OSS has to be covered by the system (or project). However, there are OSS solutions which also adhere to standards such as the Data Distribution Service (DDS) vendor Object Computing Incorporated (OCI) .

An example of an OSS offering that has suffered from a calamity is XeroMQ. Even though ZeroMQ is still around and being used, the ZeroMQ OSS effort was shaken by the death of its prime moving force Pieter Hintjens who died[227]. There are any spinoffs and derivatives of ZeroMQ. Here are a few reported in Wikipedia[228]

- *In 2012, two of the original developers forked ZeroMQ as Crossroads I/O.*[229][230]
- *Martin Sustrik has started nanomsg,*[231] *a rewrite of the ZeroMQ core library.*[232]
- *In 2012, Dongmin Yu announced his pure Java conversion of ZeroMQ, JeroMQ.*[233]
- *This has inspired further full-native ports of ZeroMQ, such as NetMQ for C#*[234]*.*
- *March 2013, Pieter Hintjens announced a new draft of the ZMTP wire-level protocol bringing*

extensible security mechanisms to ZeroMQ[235]*.*

- *Martin Hurton implemented the CurveZMQ authentication and encryption mechanism*[236] *in the core library shortly afterwards.*

Not recognizing or managing the risks due this kind of unfortunate occurrence to the system (or project) might be expedient, but is in many ways irresponsible for systems (or projects) with a long lifespan. An alternative to the risks of OSS or proprietary vendor lock-in is the selection components that are standards based from a Standards Organization that offers a wider spectrum of vendors to choose from.

DIDO Specifics

Return to the Top

To be added/expanded in future revisions of the DIDO RA

[219] Software Evolution, Blog Post, Mitopia Technologies, https://mitosystems.com/software-evolution/

[220] The History of Windows Operating Systems, Vengie Beal, Webopedia, 2 Sugust 2018, Accessed 16 July 2020, https://www.webopedia.com/DidYouKnow/Hardware_Software/history_of_microsoft_windows_operating_system.html#windows-1

[221] When will Microsoft end support for your version of Windows or Office?, Ed Bott, 10 April 2018, ZDNet, Accessed 16 July 2020, https://www.zdnet.com/article/when-will-microsoft-pull-the-plug-on-your-version-of-windows-or-office/

[222] Six Years Since World Launch, IPv6 Now Dominant Internet Protocol for Many, Internet Society, 6 June 2018, Accessed 16 July 2020, https://www.internetsociety.org/news/press-releases/2018/six-years-since-world-launch-ipv6-now-dominant-internet-protocol-for-many/

[223] IIC Connectivity Framework defines IIoT network architecture for scalable interoperability, Industrial Embedded Systems, 18 July 2020, http://industrial.embedded-computing.com/articles/iic-connectivity-framework-defines-iiot-network-architecture-for-scalable-interoperability/

[224] Tools for Distributed Systems Monitoring, Kufel, Łukasz, 1 December 2016, Foundations of Computing and Decision Sciences, Vol 41: 10.1515/fcds-2016-0014, https://www.researchgate.net/publication/311863266_Tools_for_Distributed_Systems_Monitoring/citation/download

[225] Log4j - Logging Levels, Log4J, Accessed 18 July 2020, https://www.tutorialspoint.com/log4j/log4j_logging_levels.htm

[226] Software Metrics for Predicting Maintainability, Marc Frappier, Stan Matwin, and Ali Mili, University of Ottawa, Canadian Space Agency, 1994, References 20 July 2020,

http://www.dmi.usherb.ca/~frappier/Papers/tm2.pdf

[227] The Life, Ideas, and Legacy of Pieter Hintjens (from ZeroMQ to "A Protocol for Dying"), Medium, Evan SooHoo, 23 September 2018, Accessed 20 July 2020, https://medium.com/@evan_soohoo/the-life-ideas-and-legacy-of-pieter-hintjens-from-zeromq-to-a-protocol-for-dying-fc1673caeaa7

[228] ZeroMQ, Wikipedia, Accessed 20 July 2020, https://en.wikipedia.org/wiki/ZeroMQ

[229] ZeroMQ and Crossroads I/O: Forking over trademarks. LWN.net. Retrieved 14 July 2012.https://lwn.net/Articles/488732/

[230] Crossroads I/O. Retrieved 14 July 2012, http://www.crossroads.io/

[231] nanomsg. Retrieved 8 June 2013 http://nanomsg.org/

[232] Why should I have written ZeroMQ in C, not C++ (part I) , 250bpm, Martin Sústrik, 10 May 2012, Accessed 20 July 2020, http://250bpm.com/blog:4

[233] jeromq - java pojo zeromq, zeromq-dev mailing list, Retrieved 23 May 2013, http://lists.zeromq.org/pipermail/zeromq-dev/2012-August/018265.html

[234] NetMQ, GitHub, Retrieved 23 May 2013, https://github.com/zeromq/netmq

[235] Securing ZeroMQ: draft ZMTP v3.0 Protocol, Hintjens.com, Retrieved 23 May 2013, http://hintjens.com/blog:39

[236] CurveZMQ - Security for ZeroMQ , CurveZMQ, Accessedd 20 July 2020, http://curvezmq.org/

4.3.5.4 Software Manageability Issues

Return to Manageability

About

Over the last few decades, many advances have been made in terms of Open Source Software (OSS) which has to change the way software was developed, released and used. During this time period, the traditional Waterfall Model of System and software development has also been largely supplanted with the Agile Model. Many of these changes also have to do with the evolution of systems (or projects) from being Greenfield to Brownfield development and from a "build the world" attitude towards "integrate and glue the world" mindset.

Successful OSS development and adoption not only has to produce products which are solid, strong and robust but also must meet the needs of a Community of Interest (CoI) that has coalesced around a single minded, purpose built, functionality (i.e., Apache Tomcat application server, PostgreSQL Database, Node.js an asynchronous event-driven JavaScript runtime, Docker containerized apps, Kubernetes orchestration engine for containers, etc.). Many of the OSS products are part of many of the successful projects today.

However, it is not good enough to just write software and make it publicly available. At the heart of these successful efforts are the well governed, focused, supporting CoIs. There is a desire from almost all systems (or projects) to join the OSS trend but unfortunately, the need for strong governance and rigorous methodology is minimized or skipped in the name of expediency. Fortunately, there is an organization which can help with this called Talk Openly Develop Openly (TODO) (not to be confused with a `to-do`).

TODO organization, though focused on OSS, has written a series of white papers that are well worth studying and using even it your system (or project) is not OSS. One of these papers which is particularly germane to Software Manageability is Tools for managing open source programs [24]. It is beyond the scope of this document to try to recreate the full content of this white paper. It does present a list of many of the tools available for managing software and how to use them. Here is the table of content from the document:

- Why you need special tools for open source program management
- How to select and plan your tools
- Elements of a basic toolset
- Tools for managing source code
- Tools for tracking project health
- Tools for communications and collaboration
- Tools for corporate-scale GitHub management

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[24] Tools for managing open source programs, Talk Openly Develop Openly (TODO), Accessed 20 July 2020, https://todogroup.org/guides/management-tools/

4.3.6 Usability

Return to Non-Functional Requirements

About

Usability is defined by ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models as the degree to which a product or system can be used by Stakeholder (i.e., specified users) to achieve specified goals within a specified context.

Goals

The goals of usability are[237]:

1. **Effectiveness** - The accuracy and completeness with which users achieve specified goals
2. **Efficiency** - The resources expended in relation to the accuracy and completeness with which users achieve goals.
3. **Satisfaction** - The comfort and acceptability of use.

Sub-Characteristics

This characteristic is composed of the following sub-characteristics[238]:

- ***Appropriateness Recognizability*** *- Degree to which users can recognize whether a product or system is appropriate for their needs.* (1)
- ***Learnability*** *- Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.* (1)
- ***Operability*** *- Degree to which a product or system has attributes that make it easy to operate and control.* (1)
- ***User Error Protection*** *- Degree to which a system protects users against making errors.* (1)
- ***User Interface Aesthetics*** *- Degree to which a user interface enables pleasing and satisfying interaction for the user.* (1)
- ***Accessibility*** *- Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.* (1)

See:

- ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models
- https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability
- ISO/IEC 9241-210:2019 Ergonomics of human-system interaction

Metrics

Return to the Top

Usability as a characteristic is often considered a subjective quality and left to “interpretation”, however, there are metrics which use to quantify these sub-characteristics. Before we delve into the definition of the specific metrics, it is important to understand why we need metrics rather than just rely on intuitive evaluations.

A core reason to collect Usability Metrics is to provide data about a stakeholder's understanding of a product's usability rather than the developer's understanding of usability. When the two understandings (i.e., interpretations) converge everyone is happy resulting in a way forward. That result may be to either continue in the same direction or to have a reassessment of the user's needs.

The metrics must quantify that the system meets the goals of the overall system:

1. The Effectiveness Metrics of the communication between the system and the users
2. The Efficiency Metrics of the users use of the system to accomplish their work
3. The Satisfaction Metrics of the users that the sub-characteristics of the system are met.

Ultimately, the primary objective of usability metrics for evaluating a system or product is properly engineered (i.e., neither under- or over-engineered).

- 4.3.6.1 Effectiveness Metrics
- 4.3.6.2 Efficiency Metrics
- 4.3.6.3 Satisfaction Metrics

DIDO Specifics

Return to the Top

To be added/expanded in future revisions of the DIDO RA

[237] Justin Mifsud, Usability Metrics – A Guide To Quantify The Usability Of Any System, Accessed 18 November 2020, https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/

[238] International Organization for Standardization (ISO), Usability, ISO25000, Accessed: 17 November 2020, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

4.3.6.1 Effectiveness Metrics

Return to Usability

About

There are two ways to calculate effectiveness:

- Count the success of completing tasks
- Count the errors while completing tasks

Using Task Completion Rates

Return to Top

Effectiveness measures the ratio of tasks completed versus those that are not completed. It is usually reported as a simple percentage such as 80% of the tasks are completed. The data is collected using a simple binary (i.e., yes = '1' and no = '0') for each user for each task.

Mathematically:

$$Effectiveness = \frac{\sum_{user=1}^{numberOfTesters} Completed\,Tasks\,by\,user}{\sum_{user=1}^{numberOfTesters} Total\,Tasks\,assigned\,to\,user} \times 100\,\%$$

Here is an example where the system uses 10 tasks and all the tasks are completed by 5 users. The results are:

user	Completed Tasks	Incomplete Tasks	Total Tasks	Effectiveness
user 1	8	2	10	**80%**
user 2	9	1	10	**90%**
user 3	10	0	10	**100%**
user 4	5	5	10	**50%**
user 5	6	4	10	**60%**
total	**38**	**12**	**50**	**76%**

This efficiency ratio is referred to as the **Completion Rate** or **Fundamental Usability Metric**. As a result of its simplicity and ease of understanding, the **Completion Rate** is a popular way of reporting efficiency. However, in order to determine the efficiency, a working system with real users needs to be available. This is a major advantage of using the Agile Model since the efficiency can be calculated during each Sprint and used as a guidance for future sprints.

Using Error Rates

Return to Top

Another way to understand an application or system efficiency from a usability perspective is to count the errors instead of counting the success of completing tasks. Every error can have more consequences requiring more work to “undo” thus negatively impacting Effectiveness and efficiency. The following is a partial list that users can make while completing tasks[239] :

- **Unintended actions** - using a `next` operation without intending.
- **Slips entering information** - pressing a `return` `escape` key accidentally
- **Mistakes in data entry** - twiddling the digits in a phone number or social security number or having autocorrect inadvertently “correct” an entry
- **Omissions in data entry** - forgetting login as an existing user rather than creating a new user, skipping filling in age

Recording each instance of an error along with a description can help refine the system to prevent these errors in the future. For example, “user entered last name in the first name field”.

In many ways, this metric can be considered similar to the Task Complete Rate metric but instead of counting completed tasks the errors are counted. Note, many errors can occur in accomplishing a single task. For example, a data entry form required for one task can have errors associated with each field. Each of these errors can have a negative impact on the overall efficiency.

Another enhancement to this metric wold be to assign a weight to the error. For example, data entry errors are less dramatic and important than unintended actions. Therefore, unintended actions would receive a higher weighting than data entry.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[239] Jeff Sauro, 10 Essential Usability Metrics, Measuring U, 30 November 2011, Accessed 19 November 2020, https://measuringu.com/essential-metrics/

4.3.6.2 Efficiency Metrics

Return to Usability

About

Efficiency is measured in terms of the time it takes to complete a task from when the task is initiated to when it is successfully completed. The units used to record the time must be uniform for all tasks (i.e., milliseconds, seconds, minutes, etc).

Mathematically The time taken to complete a task can then be calculated by simply subtracting the start time from the end time:

$$Task\ Time = End\ Time\ \text{-}\ Start\ Time$$

There are two ways to calculate Efficiency:

- Time-Based Efficiency
- Overall Relative Efficiency

Time-Based Efficiency

Return to Top

In this calculation, the quotient from the division of the success of a task (either one or zero) divided by the time to accomplish a task is an indicator of the efficiency of the task. For example, if a tasks was not successful, then the success of the task is zero and the efficiency is zero. If the task is successful and it takes one minute to accomplish the task, then the efficient is one (i.e., 1/1). If it takes two minutes to accomplish the task, then the efficiency is one half (i.e., 1/2 = .5).

To calculate the Time-Based Efficiency for all tasks and all users, the following equation applies:

$$Time\ Based\ Efficiency = \frac{\sum_{i=1}^{R} \sum_{j=1}^{N} \frac{n_{i,j}}{t_{i,j}}}{N \times R}$$

Where:

- **N** : The total number of tasks (goals)
- **R** : The number of users
- **$n_{i,j}$** : The result of task i by user j; if the user successfully completes the task, then Nij = 1, if not, then Nij = 0
- **t_{ij}** : The time spent by user j to complete task i. If the task is not successfully completed, then time is measured till the moment the user quits the task

Justin Mifsud[1] provides an excellent example of how for calculating time-based efficiency:

Suppose there are 4 users who use the same product to attempt to perform the same task (1 task). 3 users manage to successfully complete it – taking 1, 2 and 3 seconds respectively. The fourth user takes 6 seconds and then gives up without completing the task.

Taking the above equation:

> *N = The total number of tasks = 1*
> *R = The number of users = 4*
>
> *User 1: Nij = 1 and Tij = 1*
> *User 2: Nij = 1 and Tij = 2*
> *User 3: Nij = 1 and Tij = 3*
> *User 4: Nij = 0 and Tij = 6*

Placing the above values in the equation:

$$Time\ Based\ Efficiency = \frac{\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{0}{6}\right)}{1 \times 4} = 0.46\,\frac{goals}{sec}$$

Overall Relative Efficiency

Return to Top

The overall relative efficiency uses the ratio of the time taken by the users who successfully completed the task in relation to the total time taken by all users. The equation can thus be represented as follows[240]:

$$Overall\ Relative\ Efficiency = \frac{\sum_{i=1}^{R} \sum_{j=1}^{N} n_{i,j} \times t_{i,j}}{\sum_{i=1}^{R} \sum_{j=1}^{N} t_{i,j}} \times 100\%$$

Justin Mifsud[1] provides an excellent example of how for calculating Overall Relative Efficiency.

Assume there are 4 users who use the same product to attempt to perform the same task (1 task). 3 users manage to successfully complete it – taking 1, 2 and 3 seconds respectively. The fourth user takes 6 seconds and then gives up without completing the task.

Taking the above equation:

> *N = The total number of tasks = 1*
> *R = The number of users = 4*

User 1: Nij = 1 and Tij = 1
User 2: Nij = 1 and Tij = 2
User 3: Nij = 1 and Tij = 3
User 4: Nij = 0 and Tij = 6

Placing the above values into the equation yields the following:

$$Overall\,Relative\,Efficiency=\left(\frac{\left(\left(1\times1\right)+\left(1\times2\right)+\left(1\times3\right)+\left(1\times6\right)\right)}{\left(1+2+3+6\right)}\right)\times100=50\,\%$$

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[240] UI Designer, Efficiency, Accessed 19 November 2020, http://ui-designer.net/usability/efficiency.htm

4.3.6.3 Satisfaction Metrics

Return to Top

About

Usability Metrics are generally done through standardized questions designed to capture a the user's sentiments about the application, product or system. The survey's pose questions to the users and provide a scale of acceptability the user chooses in assessing a particular attribute. The most common scale is based on the Likert Scales originally proposed in 1032 [241].

Figure 49 gives a few of the Scales that Lickert defined. There are more available here:

Scale	Attitude / Sentiment				
Agreement	**Strongly Disagree**	**Disagree**	**Undecided**	**Agree**	**Strongly Agree**
Frequency	**Never**	**Rarely**	**Sometimes**	**Often**	**Always**
Importance	**Unimportant**	**Slightly Important**	**Moderately Important**	**Important**	**Very Important**
Quality	**Very Poor**	**Poor**	**Fair**	**Good**	**Excellent**
Likelihood	**Almost Never True**	**Usually Not True**	**Occasionally True**	**Usually True**	**Almost Always True**
Score	1	3	3	4	5

Figure 49: The Lickert Scale

There are two ways that user satisfaction can be measured:

- **Task Level Satisfaction** - The Task Level Satisfaction is made at the end of each task attempted by the user. Note, a task may be attempted but it may not be completed. Therefore, it is important to record not just the attitude or sentiment about the task, but also the status of the task when the user takes the survey.
- **Test Level Satisfaction** - Similar to the **Task Level Satisfaction**, Test Level Satisfaction is conducted at the end of a Test which can be comprised of multiple tasks. Therefore, in order to properly assess the Test Level, an evaluation of the Task assessments also needs to be made. For example, a test assessment might be low because some of the tasks were assessed as poor.

ISO also provides some guidance in how to assess User Satisfaction. See:

- ISO 10001:2018 Quality management — Customer satisfaction — Guidelines for codes of conduct for organizations
- ISO 10002:2018 Quality management — Customer satisfaction — Guidelines for complaints handling in organizations
- ISO 10003:2018 Quality management — Customer satisfaction — Guidelines for dispute resolution

external to organizations

- ISO 10004:2018 Quality management — Customer satisfaction — Guidelines for monitoring and measuring

- **Note:** For more information, see: https://blog.ansi.org/2018/07/customer-satisfaction-iso-10002-quality/#gref

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[241] Saul McLeod, Likert Scale Definition, Examples and Analysis , Simply Psychology, 2019, Accessed 20 November 2020, https://www.simplypsychology.org/likert-scale.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:30_usability:satisfaction

Last update: **2021/06/11 14:52**

4.3.7 Performance

Return to Non-Functional Requirements

About

Performance is the ability of a system to accomplish the required functionality at or under the required specification limits. The limits are generally provided relative to time (i.e., so-many transactions per second, so-many updates per millisecond, so-many recorded entries per second, etc.) The specifications can also include accuracy, precision, precision or even efficiency of other dependent systems as requirements. The following are some examples of performance specifications:

- Time for a data-entry window to appear
- Number of units produced in a given amount of time
- Time to react to a given event
- Amount of energy required to perform an activity
- The amount of computing resources required (i.e., Central Processing Unit (CPU), Random access memory (RAM), Read-Only Memory (ROM), Storage Device, Bandwidth, etc)
- Time to process a computational task such as compression-decompression, encryption-decryption, generate a Unique ID (UID), serialize-deserialize, calculate an area, calculate a new trajectory, etc.
- Time to process a storage task such as store records, index the records, retrieve records, etc.
- Time to access memory or cache such as direct memory access(DMA), the hit ratio
- Time to transfer data between components of a computer such as from memory to Central Processing Unit (CPU), from CPU to graphics card, etc

Performance can also be viewed from the following perspectives:

- 4.3.7.1 Platform Performance
- 4.3.7.2 Application Performance
- 4.3.7.3 Network Performance

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

=-
=-

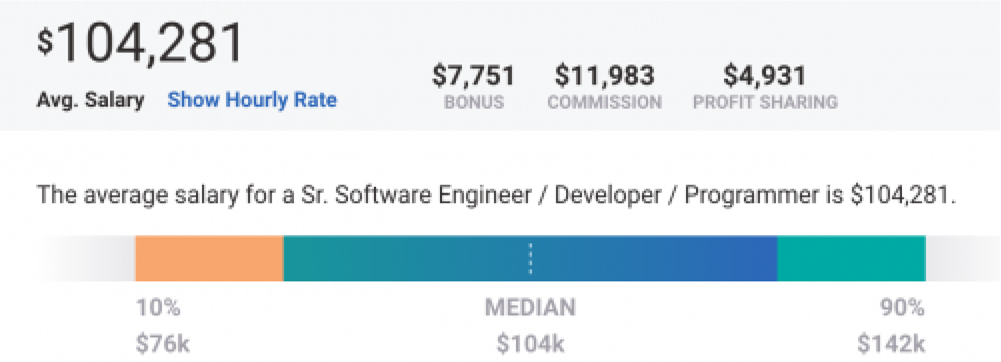
4.3.7.1 Platform Performance

Return to Performance

About

Platform Performance is concerned with the abilities of a computer system matched with an Operating System (OS) to meet or exceed the requirements of a specific system (or project). The platform could be a real system or a virtual system running locally or in the cloud. Often this comes down to the cost of the hardware and OS, but this can also include social responsibility requirements such as energy consumption, geopolitical concerns, or even ethics which prevent the use of certain products.

Figure 4 shows the salary of a Senior Software Engineer in the USA in July 2020, around $105,000. This is the take home pay, not the fully burdened cost.



$104,281

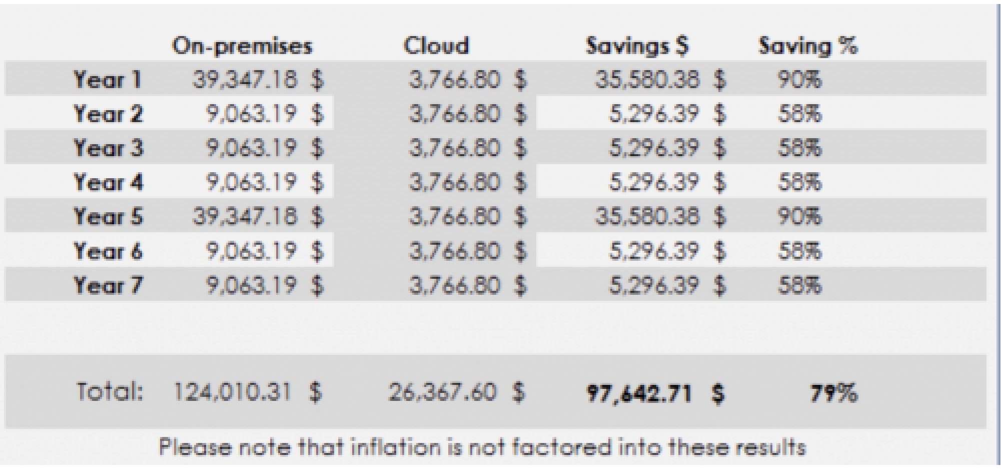
Avg. Salary Show Hourly Rate

$7,751	$11,983	$4,931
BONUS	COMMISSION	PROFIT SHARING

The average salary for a Sr. Software Engineer / Developer / Programmer is $104,281.

Figure 4: Average Sr. Software Engineer / Developer / Programmer Salary - June 2020[25]

Figure 5 shows the cost of either owning a Server or using a Infrastructure-as-a-Service (IaaS) solution. The Total Cost of Ownership (TCO) cost is about 10 times that of an IaaS.



	On-premises	Cloud	Savings $	Saving %
Year 1	39,347.18 $	3,766.80 $	35,580.38 $	90%
Year 2	9,063.19 $	3,766.80 $	5,296.39 $	58%
Year 3	9,063.19 $	3,766.80 $	5,296.39 $	58%
Year 4	9,063.19 $	3,766.80 $	5,296.39 $	58%
Year 5	39,347.18 $	3,766.80 $	35,580.38 $	90%
Year 6	9,063.19 $	3,766.80 $	5,296.39 $	58%
Year 7	9,063.19 $	3,766.80 $	5,296.39 $	58%
Total:	124,010.31 $	26,367.60 $	**97,642.71 $**	**79%**

Please note that inflation is not factored into these results

Figure 5: Average Total Cost of Ownership (TCO) versus Cloud [26]

If your application is not performing well and it is estimated that it might take one year to upgrade the software, you could get a senior Software Engineer for the $105,000 or you could upgrade your server. If it is a purchase, you could get almost three servers for the cost of an engineer. If you decided to use IaaS, for the cost of the engineer, you could get almost 20 servers.

It is recommended to follow the guidelines for sizing a server provided on-line.[27][28][29]

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[25] Payscale.com, Accessed 20 July 2020, https://www.payscale.com)

[26] Cost of server ownership: on-premise vs. IaaS, April 2019, Sophie Furnival, Sherweb, Accessed 20 July 2020, https://www.sherweb.com/blog/cloud-server/total-cost-of-ownership-of-servers-iaas-vs-on-premise/

[27] Basic Guidelines for Sizing Servers, Jason Thomas, 24 August 2017, Accessed 20 July 2020, https://www.mirazon.com/basic-guidelines-for-sizing-servers/

[28] 3 secrets to right-sizing a server, TidalScale, Accessed 20 July 2020, https://www.tidalscale.com/3-secrets-to-right-sizing-a-server/

[29] Determining Your Ideal Server Size: Which Package is Right for Me?, Media Temple, 7 April 2015, Accessed 20 July 2020, https://mediatemple.net/resources/web-hosting-101/determining-your-ideal-server-size-which-package-is-right-for-me/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:40_performance:01_platform

Last update: **2021/06/14 17:53**

4.3.7.2 Application Performance

Return to Performance

About

Application Performance are measures of real-world performance and availability of applications. It is often used to describe remote and cloud computing applications, however, there is a lot of time spent on tuning applications running locally to enhance or improve their performance. For example file access time, database, infrastructure, graphics, human interfaces, network access, etc.

Most applications today are not Brownfield versus Greenfield systems. In addition, many programs rely on incorporating reusable components as part of the cost reduction strategy. Reusing components such as operating systems, databases, Access Control etc. makes a lot of sense since the functionality of these components is not an area of expertise for the developers of the system. For example, if the system is for medical devices, then designing new virtual page swap software has nothing to do with medical technology or medicine. Another example might be developing middleware software for accessing other computers on the network. This kind of development might seem “fun” but it is not within the scope of the project that is developing the system. These re-used components also have the benefit that they are used by a wide range of other projects and systems accessed by many people. Collectively the communities can find and fix software faster than it can be done locally.

I don't believe there is a company that could pay for all the testing the user communities provide. Linux is a great example.[247)]

- *About 3 to 3.5 billion, which is pretty much the number of humans on the planet who have regular access to the internet. Let’s break it down….*
- *Directly as a desktop OS? A minority of users, sadly windows is hogging over 85% of that market share - which it certainly doesn’t deserve. The remaining 10 to 15 percent is mostly Apple, who have done a very good job at creating its desktop OS based on a Unix implementation, so it’s kind of a “cousin” of Linux in that sense.*
- *If you use an Android phone or tablet, you’re using the Linux kernel and GNU software indirectly. Many Android devices will show this in the “kernel info” or “kernel version” in the settings app’s About section.*
- *If you have a router, modem, printer, scanner, or any similar device, its firmware is probably based on a small Linux or FreeBSD system.*

It is also true that this many users also marks Linux as a huge target for malicious activities, but pretending that any OS is not vulnerable to attack is a fool's game.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[247] Éric Nunya, 9 March 2020, Quora, ccessed 27 July 2020, [https://www.quora.com/How-many-Linux-users-are-there-in-the-world[]]

4.3.7.3 Network Performance

Return to Performance

About

Return to Top

Network Performance captures the statistical metrics and the analytical review of a network. Collectively they reflect the network's Quality of Services.

It is a qualitative and quantitative process that measures and defines the performance level of a given network. It guides a network administrator in the review, measure and improvement of network services.

There are two main ways to connect devices together:

- Ethernet (i.e., wired using network cables.)
- Wireless Fidelity (Wi-Fi) (i.e., wireless using radio signals).

Although it is possible to connect computers directly together, generally the computers connect to a Network Device such as a Router. There are a number of variables that determine the actual speed of the connection between the computers. The wired connections are as a rule faster than the wireless connection but the number, length of the quality of the network connections and the kinds of network devices and the number of devices can impact wired connection speed. WiFi connections are more susceptible to interference from electrical devices, physical objects (i.e., metal walls or cages), or environmental conditions (i.e., weather and solar flares).

An Ethernet connection is consequently more reliable especially when Shielding Network Cabling (i.e., Category 6 (Cat-6),Category 7 (Cat-7), Category 8 (Cat-8)) are used. Ethernet is almost always faster than WiFi. The fastest Ethernet speeds today top out at 10Gbps or higher, while the fastest WiFi speeds theoretically max out at 6.9Gbps, though actual speeds are much slower – usually less than 1Gbps.[248].

Speed

Return to Top

Network speed is for the most part about acquiring the correctly sized physical Network Devices (i.e. Modem, Router, Switches, Network Cabling, etc.) to meet the demands of the system. However, there are restrictions that arise such as the need for wireless connections (i.e., WiFi, Bluetooth, ZigBee, Infrared Wireless Networking etc.), space and heat considerations (i.e., a big problems for planes, ships, labs, hospital rooms, etc.) or when the assets participating in the system are distributed and not under the control of a single source (i.e., blockchains, Distributed Ledger Technologies (DLT), supply chains, etc.).

Most Ethernet connections fall into the following categories: Wired Network and Wireless Network

The wired networks use hardware such a modems, routers, switches, cabling, etc. together.

Wireless cables connect to each and every one of the computers in the network. The cost of a wired network is lower compared to the wireless network since Ethernet, cables, and switches are not expensive. Wired LAN offers better performance compared to wireless networks.

Wired Connections

Return to Top

Table 20: Summary of Difference between CAT-5 through CAT-8[249]

Category	Standard	Data rate	Frequency[1]	# of Conductors
Category 5 (Cat-5)	100BASE-TX	100Mbit	100 Mhz	4 or 8
Cat-5E	100BASE-TX	1Gbit	100 Mhz Duplex	8
Category 6 (Cat-6)	EIA 568B2.1	1-10 Gbit[2]	250 Mhz	8
Cat-6A	10GBASE-T	10 Gbit	500 Mhz	8
Category 7 (Cat-7)	10GBASE-T	10 Gbit	600 Mhz	8
Cat-7A	10GBASE-T	10 Gbit	1000 Mhz	8
Category 8 (Cat-8)	40GBASE-T	40 Gbit	1600-200Mhz Mhz	8

[1] **Note:** 1 hertz is roughly equivalent to 1000 milliseconds, 20 killohertz is roughly equivalent to 0.05 milliseconds. See Unit Juggler.
[2] **Note:** Depends on the length and cable type

Wireless Connections

Return to Top

Table 21: Side-by-side comparison of wireless routers[250]

	Netgear Nighthawk X10 AD7200	Asus RT-AC86U AC2900	Linksys EA6350 AC1200+	TP-Link Archer C7 AC1750	Trendnet AC2600	TP-Link AC2300	Linksys WRT32X
Top Theoretical Speed	4600 Mbps on 60 GHz	2167 Mbps on 5GHz	867 Mbps on 5 GHz	1300 Mbps on 5 GHz	1733 Mbps on 5 GHz	1625 Mbps on 5 GHz	2600 Mbps on 5 GHz

Bandwidth

Return to Top

Bandwidth is defined as the bandwidth data carrying capacity of a network channel or the entire network. Bandwidth is measured by the bit-rate of the network transmission capacity. This is sometimes thought of as the network channel's data transfer speed. Bandwidth can be used to describe wired, wireless or even data buses.

A bit represents a single binary digit either '0' or '1'. The '0' or '1' generally represent yes/no, true/false, on/off, or up/down/ It does not necessarily equate a '0' with false and a '1' with true. When transmitted over a network, the data is sent as a stream of bits (not bytes). A byte is generally used to signify a unit of memory or storage (i.e., RAM or ROM) that usually is eights bits long (wide) and is the smallest number of bits used to represent a character in the original ASCII character set used by most most computers.

Bandwidth is measured as bits per second and is used as a denominator of bits (i.e., kilobits, megabits). When bandwidth is used to describe a network connection (i.e., switch, server or router), it is generally in megabits, however, when it is used to describe the data flowing into the connection then bandwidth referred to as traffic and could be measured in either megabits per second (Mb/s or Mbps) or megabytes per second (MB/s or MBps). Although the nomencalture is subtle, it is important to be aware of the difference. An inadvertent misunderstanding could result in a error of magnitude 8 (i.e., 1 byte = 8 bits).

Since the megabytes figure will be larger than the megabits figure (equation to follow shortly) most industry service providers like to give a total transfer based on this figure – however most bandwidth providers use megabits.

$$1\, byte = 8\, bits$$

Some examples:

Table 22: Some examples of converting MegaBits to MegaBytes

Mega Bytes per second	bits per Byte	MegaBits per second
8 MBps	*8	64 Mbps [1]
9 MBps	*8	72 Mbps
10 MBps	*8	80 Mbps
20 MBps	*8	160 Mbps

[1] **Note:** there are two kinds of units listed: Mbps (Mega BIT per second) and MBps (Mega BYTE per second)

> *Today, many cable ISPs are capable of delivering internet speeds over 1 Gigabit per second. That's 1 billion bits per second! Not everyone needs this much speed today (Netflix reports that a connection speed of 25 Megabits per second is all that's required to stream Ultra HD content), but cable ISPs see a future of virtual reality, telehealth, driverless cars, and an internet of things. In that environment, speed requirements are going to increase. Regardless of whether it's necessary today, ISPs are preparing their networks for the needs of the future. So, while we'll likely always measure speed in bits and data volume in bytes, the consistency and speed at which those bits are delivered over the internet will surely rise.*[251]

In addition to the Internet Service Provider (ISP) bandwidth limitations, it is also important to remember that there are local hardware components to the network. For example, Network Quality of Service (QoS) Policies.

Network Quality of Service (QoS)

Return to Top

Quality of service (QoS) captures the metrics used to measure the overall performance of a computer network as experienced by participants (i.e., computers, processes, devices, etc) in network.

Internet Protocol (IP) networks, QoS is particularly focused on setting priorities for packet traffic and reserving resources rather than the QoS of the Network Services (i.e, DNS). It helps establish overall priorities for applications, users, or data flows, or to guarantee a certain level of performance to a data flow. For example, setting Voice over IP (VOIP) as a priority over texting.[252] Some of the common metrics used to describe Network QoS are:

- Throughput
- Latency
- Packet Loss
- Jitter
- Bit Error
- Download Speed
- Upload Speed

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[248] What is the difference between a WiFi and Ethernet connection?, Spectrum Enterprise, Accessed 27 July 2020, https://enterprise.spectrum.com/support/faq/network/what-is-the-difference-between-wifi-and-ethernet-connection.html

[249] How to Tell CAT Data Cables Apart, Horst Messerer, Machine Design, 8 January 2016, Accessed 24 July 2020, https://www.machinedesign.com/automation-iiot/cables-connectors-enclosures/article/21834690/how-to-tell-cat-data-cables-apart

[250] Connect at Speed with the Fastest Routers, Speedtest, PCMag, Accessed 21 July 2020, https://www.speedtest.net/about/knowledge/fastest-routers

[251] Why Do We Use Bits to Measure Internet Speed but Bytes to Measure Data?, The Internet and Television Association (NCTA), 21 July 1017, Accessed 20 July 2020, https://www.ncta.com/whats-new/why-do-we-use-bits-measure-internet-speed-but-bytes-measure-data

[252] Andrew Froehlich, The Basics Of QoS, 15 August 2016, Accessed 27 July 2020,

https://www.networkcomputing.com/networking/basics-qos

4.3.8 Interoperability

Return to Non-Functional Requirements

About

https://www.isko.org/cyclo/interoperability#3.1

Interoperability is a characteristic of a product or system whose interfaces are completely understood to work with other products or systems, present or future, in either implementation or access, without any restrictions. There are different levels of interoperability. As one moves up the interoperability levels, the degree of interoperability increases in difficulty. The most difficult is the semantic level, in which two assets can communicate with little or no prep work using common detailed formal, machine readable Ontologies including a vocabulary of terms.

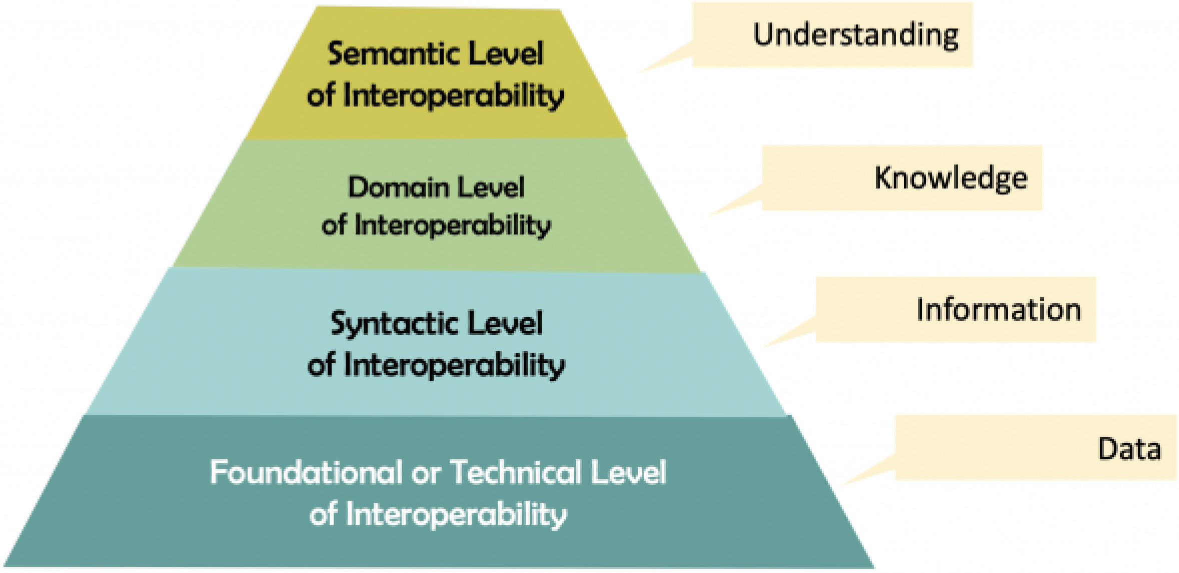

Semantic Level of Interoperability
Understanding
Domain Level of Interoperability
Knowledge
Syntactic Level of Interoperability
Information
Foundational or Technical Level of Interoperability
Data

Figure 2: The Automation Pyramid and Interoperability

Foundational or Technical Level

return to Top

The *Foundational* or *Technical Interoperability Level* is concerned with the most fundamental and basic kind of interoperability. It provides the “foundation” for all the higher levels of interoperability. It establishes the technical aspects of peer-to-peer interoperability that prevail regardless of hardware, operating system, or middleware platforms. The Foundational or Technical Interoperability Level is defined in terms of two perspectives: Data and System.

- From the Data perspective, there is a clear, shared expectation for the contents, context and meaning of that data (e.g., Big-Endian or Little-Endian, or size of an integer, or character set used)
- From the system perspective the differences in computer technology are non-existent (e.g., 16-Bit, 32-Bit and 64-Bit processors. Reduced Instruction Set Computer (RISC) versus Complex Instruction Set Computer (CISC) computers, etc.). In a Distributed System, the network protocols are another part of the Foundational or Technical Level of interoperability (e.g., Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) over Internet Protocol address (IP address), etc.).

Syntactic Level

return to Top

The *Syntax Level* is concerned with the correct combination and sequence of the elements in a language and, in this context, as it applies to Data Structure or a Programming Language. For example, in English, we do not say `ball red large plastic`, we would say `large, red plastic ball`. The first form is syntactically incorrect, the second form is not. Interoperability at the syntactical Level means two different systems can exchange information and be structurally equivalent. The interpretation of the syntactical data can be different (i.e., big to an ant is different than big to an elephant). The difference in interpretation is a semantic difference (see Semantic Level).

Domain or Structural Level

return to Top

The *Domain Level* is knowledge of a specific, specialized subject area, discipline or field. For example, there is specific domain knowledge in Chemistry, Organic Chemistry, Physics, Mathematics, or Statistics. A specific example might be the use of the word `round`. Round has different meanings in the context of Poker, Math, Cheese, and the Military. This is in contrast to general knowledge, or domain-independent knowledge which, in general, refers to the common usage of a word, sometimes referred to as the dictionary usage of a word. The use of the word Domain or Domain knowledge is used within general purpose disciplines such as history, law, systems engineering or even computer science. People with knowledge in a generic area (i.e., law) might have specific domain knowledge in computers, electronics, computers or even specific computer architectures such as Reduced Instruction Set Computer (RISC).

Semantic Level

return to Top

The *Semantic Level* is concerned with the meaning, relationship and restrictions on data and information described in the Domain or Structural and Syntactical Levels. At this level, computer systems exchange information unambiguously. The information exchanged might not be identical but there is a shared understanding between the two systems about the meaning of the information. Some examples of

Semantic Interoperability are described by the ability of:[6]:

- Two independent programs with reasoning capability to arrive at the same conclusions from the same data
- An information system, without additional human intervention, to perform the tasks for which it was designed through the exchange of relevant data with other systems[7], independently of how and what purpose this data was originally collected, stored and managed.
- One system to use data from an external source (without prior planning) and still able to output useful (though not necessarily perfect) results.

DIDO Specifics

return to Top

To be added/expanded in future revisions of the DIDO RA

[6] Ontology Taxonomy Coordinating WG / OntacGlossary, Accessed 10 July 2020, http://colab.cim3.net/cgi-bin/wiki.pl?OntologyTaxonomyCoordinatingWG%2FOntacGlossary

[7] Note: mediated by formally specific definitions and axioms

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:05_interoperability

Last update: **2021/06/11 14:36**

4.3.9 Elasticity

Return to Non-Functional Requirements

About

Cloud Elasticity also known as Elasticity “is the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible.”[255]. A primary motivation behind Elasticity is to save money by not investing in Infrastructure-as-a-Service (IaaS) that is not used or under used. It also saves natural resources since heating and air conditioning are not used on resources that are on standby[256].

The following are the various strategies used to achieve elasticity:

- **Cost-aware criteria**: The default is to assume that there is a firm fixed price for IaaS providers, however, some providers allow for spot pricing schemes (i.e., Amazon) which can allow users to tap into IaaS excess capacity. This excess capacity is there so that the IaaS provider can meet the Service Level Agreements (SLAs) guaranteed to all customers.
- **Power-aware cost function**: Using the power required to meet the application's needs and little more, i.e., using off-peak power consumption only.
- **Multiple classes of requests**: Allow applications to be segmented into categories based on the need for service. For example, customers' requests for service from the application can be divided into three categories: High Priority for performing financial transactions; Medium Priority for those making product inquiries; Low priority for simple browsing.
- **Scaling multiple applications**: Allow an application to be broken up into smaller applications whose functionality and services are orchestrated.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[255] Nikolas Roman Herbst, Samuel Kounev and Ralf Reussner, Elasticity in Cloud Computing: What It Is, and What It Is Not, Accessed on 11 August 2020, https://sdqweb.ipd.kit.edu/publications/pdfs/HeKoRe2013-ICAC-Elasticity.pdf

[256] Rui Han, Investigations into Elasticity in Cloud Computing, November 2013, Accessed 12 August 2020, https://arxiv.org/ftp/arxiv/papers/1511/1511.04651.pdf

4.3.10 Scalability

Return to Non-Functional Requirements

About

Scalability is the ability of a system to accomplish more work while maintaining the quality (i.e., without degradation) of the products produced or services provided by the system. There are different ways to calculate the work produced or performed by the system, which usually depends on the kind of product produced or services rendered.

For software systems, here are some of the common metrics used to quantify the products produced or the services provided:

- Number of transactions or events per unit of time (e.g., 5,000 credit card transactions a second[257])
- Number of tweets processed per unit of time (e.g., 6,000 tweets a second[258])
- Number of hours of videos uploaded per minute (e.g., 500 hours of fresh video per minute[259])
- Number of searches per day (e.g., 3.5 billion searches per day[260]).

Scalability is about being able to increase the output of products and services without major disruptions, interruptions or increased costs. Often, because of the Economies of Scale, the estimates for the costs should actually decline.

Two valid approaches to achieve Scalability are:

- Scaling Up: An approach generally applied to centralized or decentralized systems or products. It amounts to just adding more resources with more powerful versions. This method works until you have reach the limits on the availability of more sources power, for example, the speed of the Central Processing Unit (CPU), the amount of available memory or even the network capacity or speed. In decentralized systems(i.e., Cloud Computing) scaling can be either Vertical or Horizontal.
- Scaling Out: An approach usually associated with a Distributed System, which by its nature, allows for more Network Nodes, replicated with prepacked applications (i.e., Distributed Application (ĐApp or DApp)), that can be added with minimal overhead cost. In essence, adding more nodes offering redundant products and services. Alternatively, one can divide up the problem by functionality. For example, if accessing customer data is the bottleneck, then add more nodes with which to access the same data. If access to the actual data is the bottleneck, adding replications to the data store is recommended.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[257] Ryan Vlastelica, Why bitcoin won’t displace Visa or Mastercard soon Market Watch, 18 December 2017, Accessed 10 August 2020, https://www.marketwatch.com/story/why-bitcoin-wont-displace-visa-or-mastercard-soon-2017-12-15

[258] Kit Smith, 60 Incredible and Interesting Twitter Stats and Statistics, Bandwidth, 2 January 2020, Accesssed 10 August 2020, https://www.brandwatch.com/blog/twitter-stats-and-statistics/

[259] James Hale, More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute, Tubefilter, 7 May 2019, Acessed 10 August 2020, https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/

[260] Maryam Mohsin, 10 Google Search Statistics You Need to Know in 2020 [Infographic], Oberlo, 3 April 2020, Accessed 10 August 2020, https://www.oberlo.com/blog/google-search-statistics

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:2_nonfunc:16_scalability

Last update: **2021/06/11 14:48**

4.4 Assessing Requirements

Return to Requirements

An important part of having functional and non-functional requirements is access a particular effort to determine if the effort complies with the requirements. The approach is slightly different between Functional and non-functional requirements.

For Functional Requirements, it is important to think about what requirements need to be covered for a particular effort. For example, which Networks need to be supported? (Wired Network, Wireless Network. Bluetooth, ZigBee, Near-Field-Communication (NFC), etc).

Non-functional Requirements often have a large impact on the components that are selected as part of the infrastructure. For example, if a component is not scalable, the product built using that component is probably also not scalable.

The following sections are meant as aids in helping evaluate a project Functional and Non-FUnctional Requirements.

- 4.4.1 Functional Requirements Assessment
- 4.4.2 Non-functional Requirements Assessment

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:1.4_req:3_assessment

Last update: **2021/06/12 18:38**

4.4.2 Non-functional Requirements Assessment

Return to Assessment

- Link to the DIDO-RA Google WorkBook

Assessing the Alternatives

Return to Top

Non-Functional requirements are by nature, hard to measure and hard to assess for compliance. The complexity of the assessment problem is compounded because products or systems as well as the environment they operate within are not static. Therefore, an assessment that is done today, may no longer be accurate in the future. When assessing Commercial Off-The-Shelf (COTS), Government Off-The-Shelf (GOTS), Modified Off-The-Shelf (MOTS) or NATO Off-The-Shelf (NOTS) or the inclusion of Open Source Software (OSS) the potential for changes (particularly enhancements) needs to be assed also.

One way to accomplish this is to provided an assessment which is weighted for the “ease of implementation” for new features. For example, a system or product may not have done much to support 4.3.4.1 Confidentiality, but the vendor of the product must determine that it is an easy upgrade to add it to the product. On the other hand, the support for **Confidentiality** might be extremely difficult. Sometimes, the feature may be easy to solve but requires time and money to accomplish. As a potential stakeholder, they can direct resources to help overcome this shortfall.

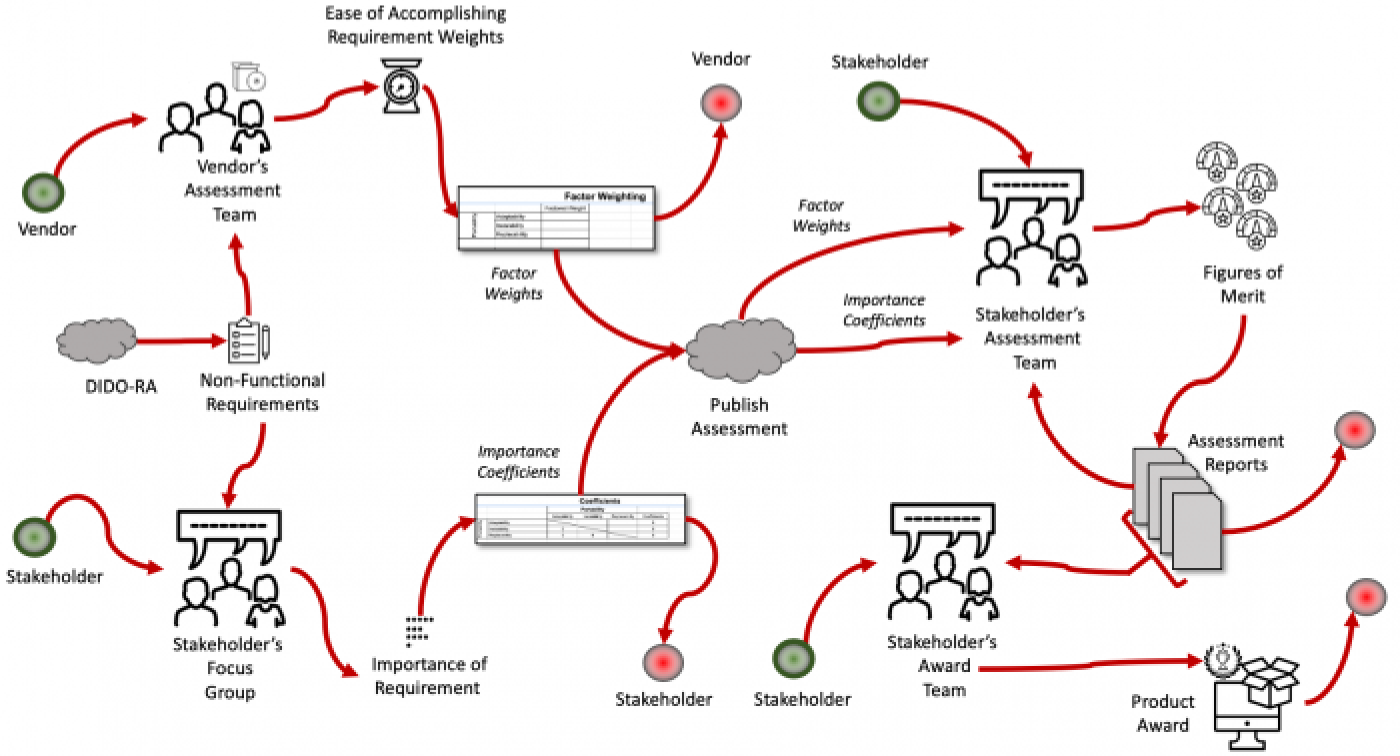


Figure 53: An overview of the Assessment Process

The Vendor's Assessment

Return to Top

The Vendor's assessment team must discuss with each other and use the DIDO-RA workbook to determine the weight of the requirement.

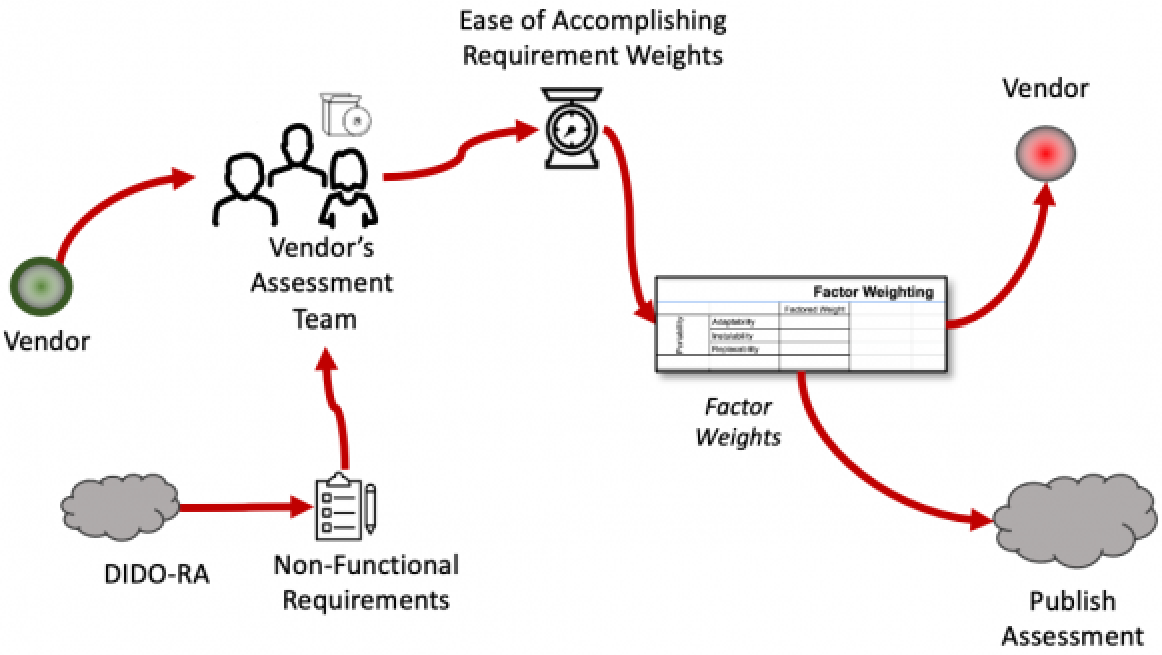
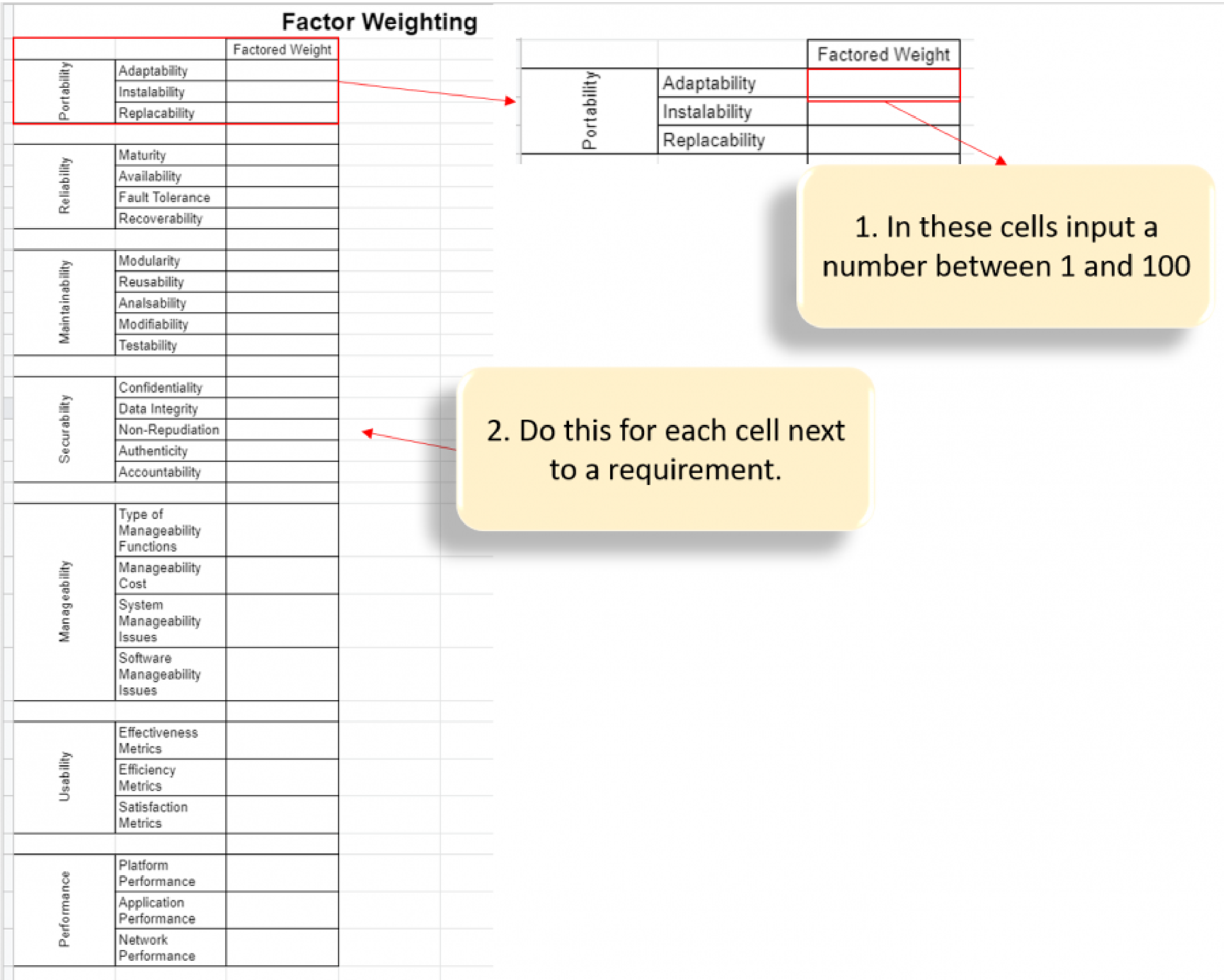


Figure 54: The Vendor's Assessment Team Activities

This is done by weighing each property, related to the Requirement, by giving a score between 1-100. 1 representing an easy development process, and 100 being impossible now, merely very difficult in a couple years.



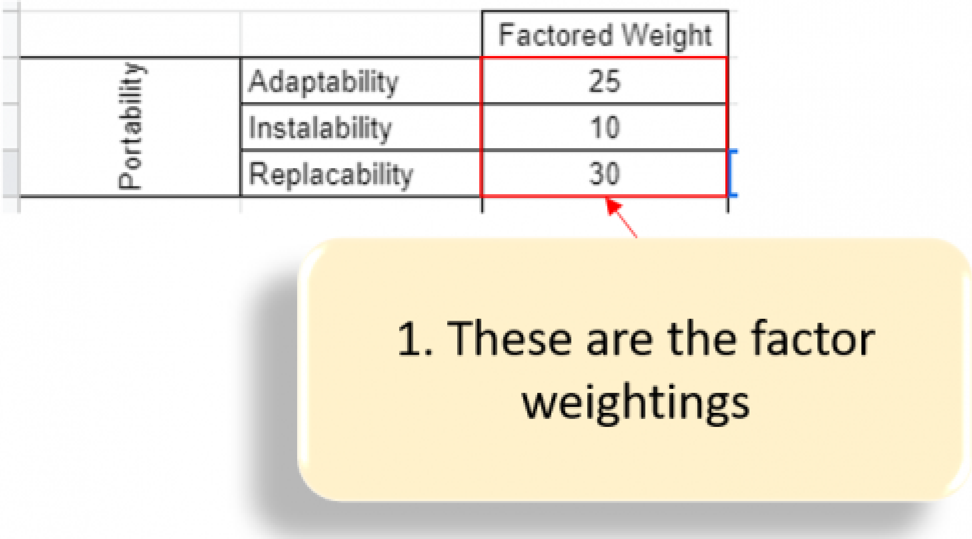
Factor Weighting

		Factored Weight
Portability	Adaptability	
	Instalability	
	Replacability	
Reliability	Maturity	
	Availability	
	Fault Tolerance	
	Recoverability	
Maintainability	Modularity	
	Reusability	
	Analsability	
	Modifiability	
	Testability	
Securability	Confidentiality	
	Data Integrity	
	Non-Repudiation	
	Authenticity	
	Accountability	
Manageability	Type of Manageability Functions	
	Manageability Cost	
	System Manageability Issues	
	Software Manageability Issues	
Usability	Effectiveness Metrics	
	Efficiency Metrics	
	Satisfaction Metrics	
Performance	Platform Performance	
	Application Performance	
	Network Performance	

		Factored Weight
Portability	Adaptability	
	Instalability	
	Replacability	

1. In these cells input a number between 1 and 100

2. Do this for each cell next to a requirement.



		Factored Weight
Portability	Adaptability	25
	Instalability	10
	Replacability	30

1. These are the factor weightings

Publish the assessment, and notify the vendor..

The Stakeholder's Focus Group

Return to Top

The Vendor's stakeholder assessment team must create a focus group of stakeholder's. This group is used in conjunction with the stakeholder assessment team to determine the importance of the Requirement and potentially direct more or less resources to this area.

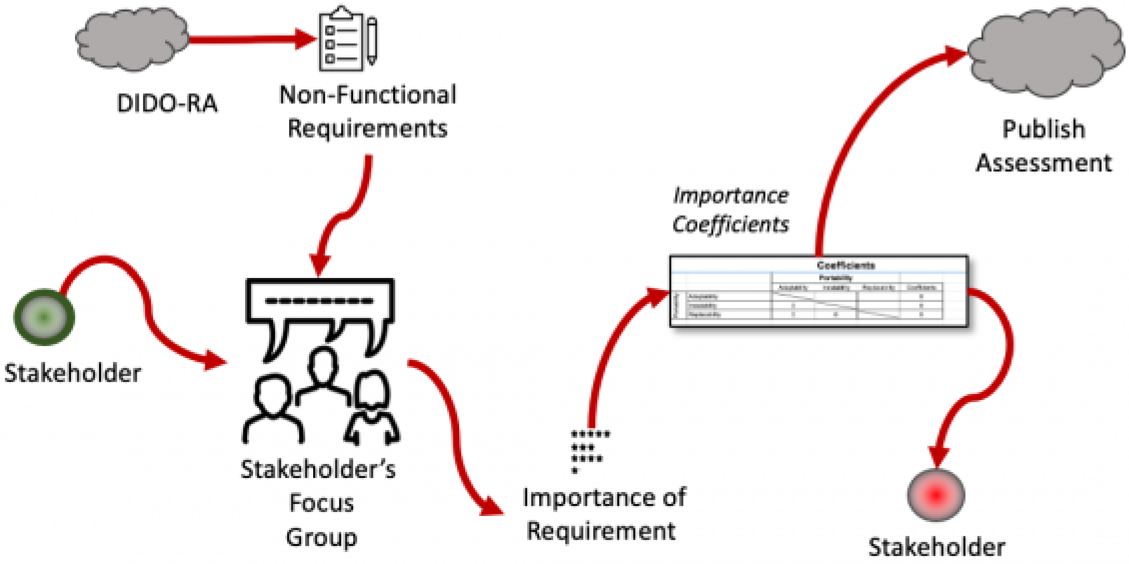
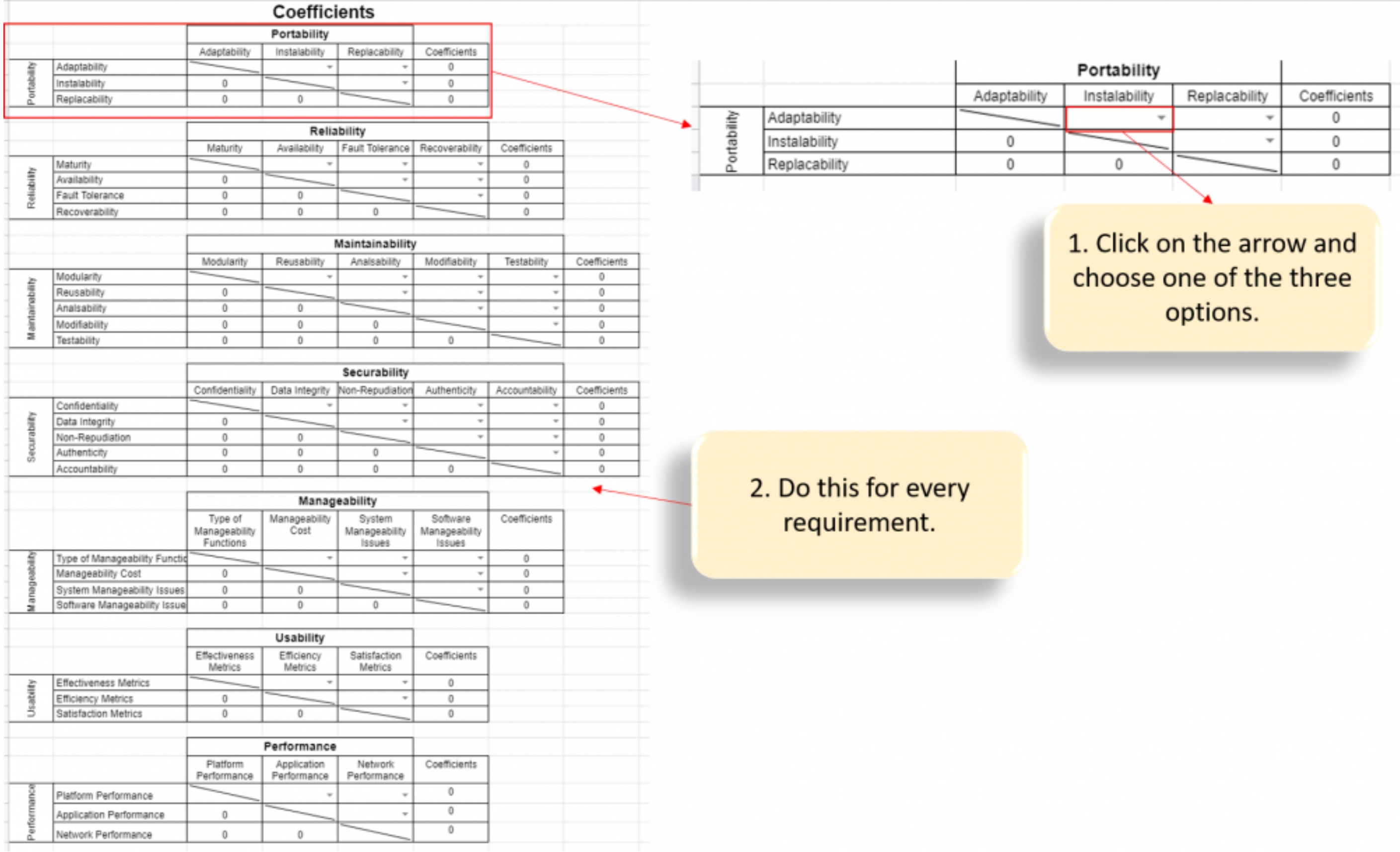


Figure 57: The Stakeholder's Focus Group Activities

This is done by rating each property related to the requirement with either a '+' option representing a 'more important' status, or a '-' option representing a 'less important' status. On the Coefficient sheet choosing a '+' will cause a '-' to be put in the opposite cell in the table. EX: cell(1,2)='+' means cell(2,1)='-'. Also properties can't be compared to themselves, these cells have lines drawn through them to represent this.



Coefficients

		Portability			
		Adaptability	Instalability	Replacability	Coefficients
Portability	Adaptability				0
	Instalability	0			0
	Replacability	0	0		0

		Reliability				
		Maturity	Availability	Fault Tolerance	Recoverability	Coefficients
Reliability	Maturity					0
	Availability	0				0
	Fault Tolerance	0	0			0
	Recoverability	0	0	0		0

		Maintainability					
		Modularity	Reusability	Analsability	Modifiability	Testability	Coefficients
Maintainability	Modularity						0
	Reusability	0					0
	Analsability	0	0				0
	Modifiability	0	0	0			0
	Testability	0	0	0	0		0

		Securability					
		Confidentiality	Data Integrity	Non-Repudiation	Authenticity	Accountability	Coefficients
Securability	Confidentiality						0
	Data Integrity	0					0
	Non-Repudiation	0	0				0
	Authenticity	0	0	0			0
	Accountability	0	0	0	0		0

		Manageability				
		Type of Manageability Functions	Manageability Cost	System Manageability Issues	Software Manageability Issues	Coefficients
Manageability	Type of Manageability Functic					0
	Manageability Cost	0				0
	System Manageability Issues	0	0			0
	Software Manageability Issue	0	0	0		0

		Usability			
		Effectiveness Metrics	Efficiency Metrics	Satisfaction Metrics	Coefficients
Usability	Effectiveness Metrics				0
	Efficiency Metrics	0			0
	Satisfaction Metrics	0	0		0

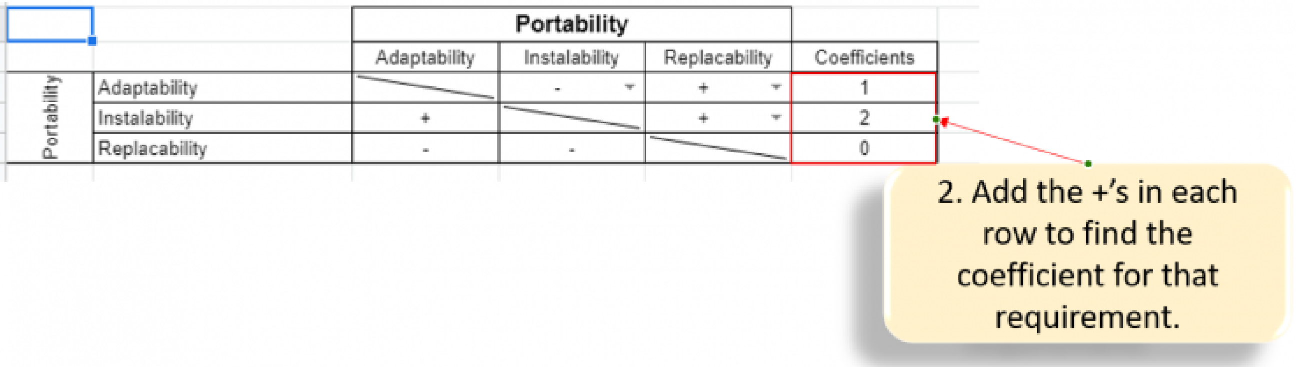
		Performance			
		Platform Performance	Application Performance	Network Performance	Coefficients
Performance	Platform Performance				0
	Application Performance	0			0
	Network Performance	0	0		0

		Portability			
		Adaptability	Instalability	Replacability	Coefficients
Portability	Adaptability				0
	Instalability	0			0
	Replacability	0	0		0

1. Click on the arrow and choose one of the three options.

2. Do this for every requirement.

At the end of filling out each table. Each row is counted for '+', the number of '+' in a row is equal to the coefficient for that property.



		Portability			
		Adaptability	Instalability	Replacability	Coefficients
Portability	Adaptability		-	+	1
	Instalability	+		+	2
	Replacability	-	-		0

2. Add the +'s in each row to find the coefficient for that requirement.

When completed publish the assessment and notify the stakeholder's.

The Stakeholder's Assessment Team

Return to Top

Once the assessments and importance coefficients are set, the DIDO-RA is passed to the stakeholder's assessments team. Here they must take the factor weighting and the coefficients to create the Figure of Merit (FoM). These figures are then published to assessment reports that are given to the Vendor, and/or given back to the stakeholder's assessment team for further review.

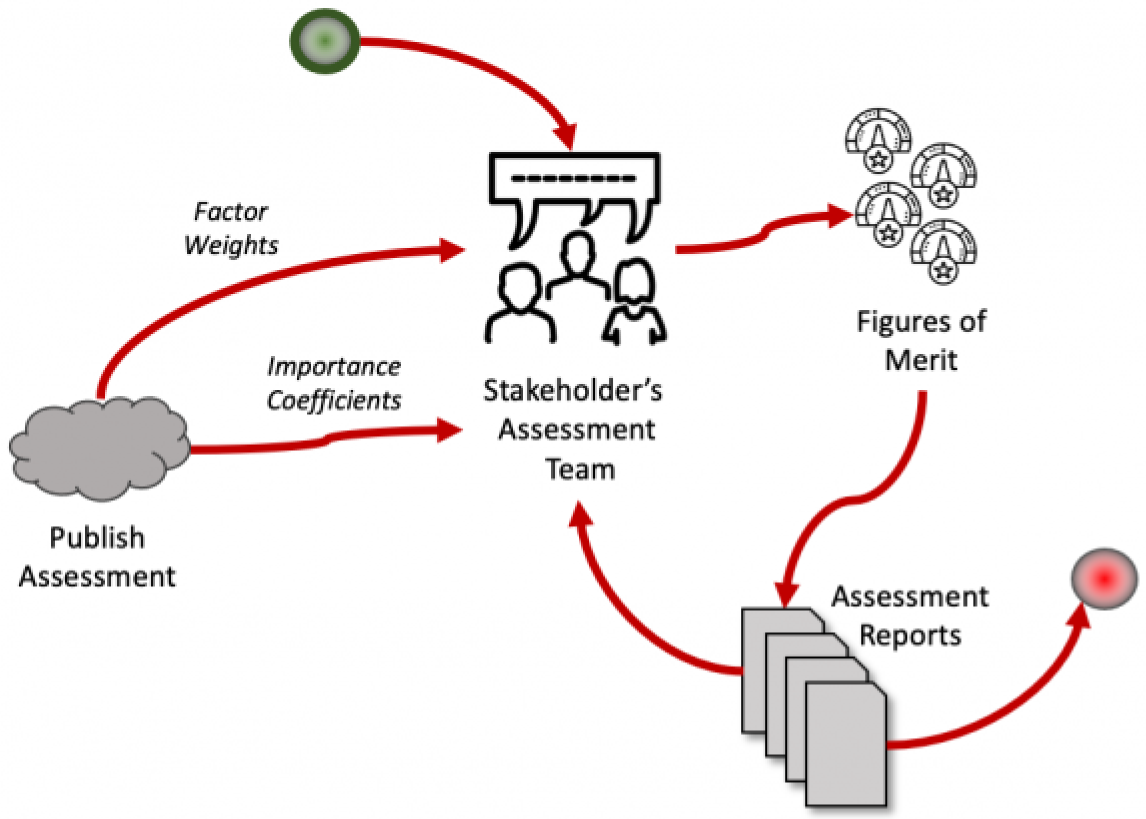
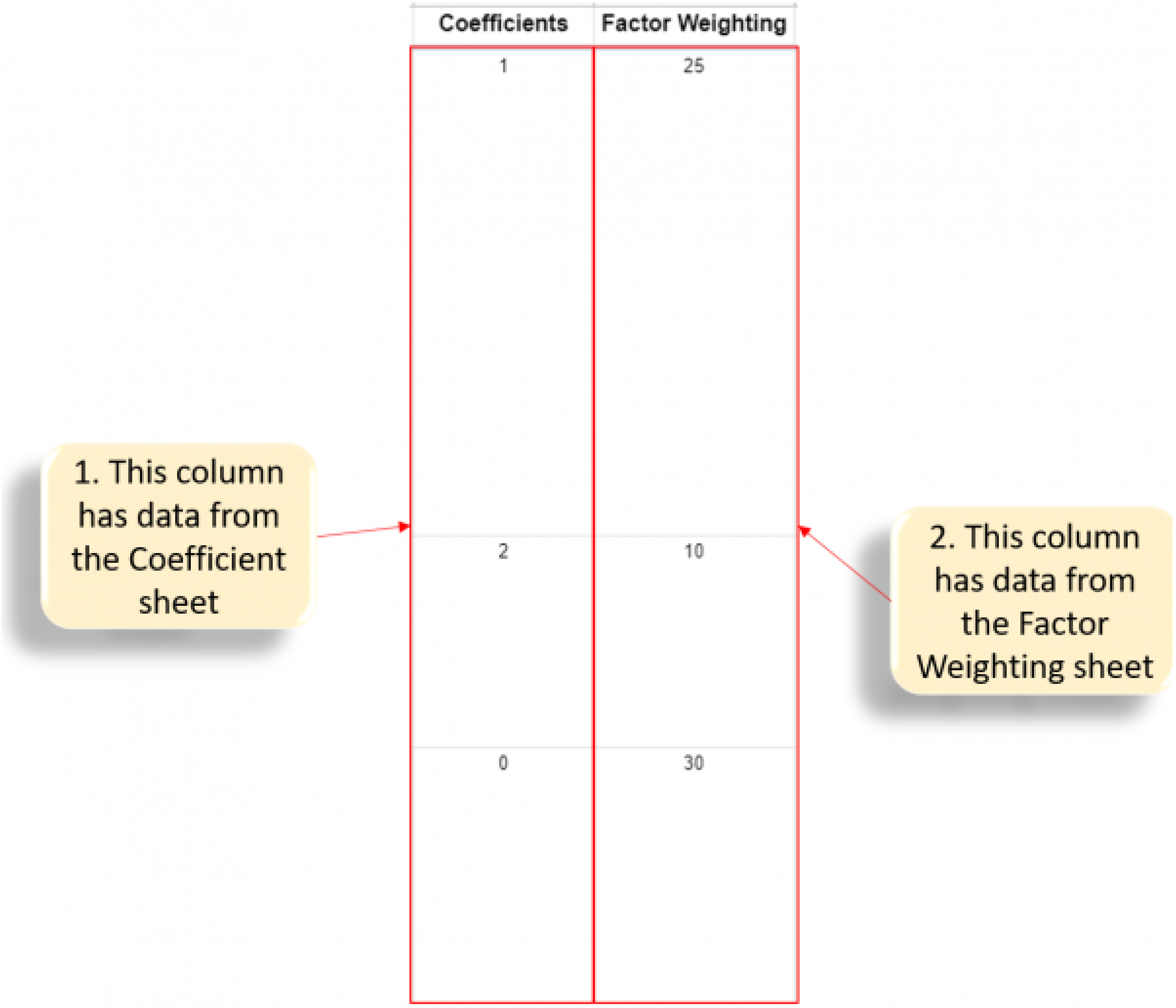


Figure 60: The Stakeholder's Assessment Team Activities

In the DIDO-RA workbook after the Coefficient/Factor Weight sheets are filled out, those values associated with the requirement, are then sent to the corresponding Requirement Sheets in the workbook.

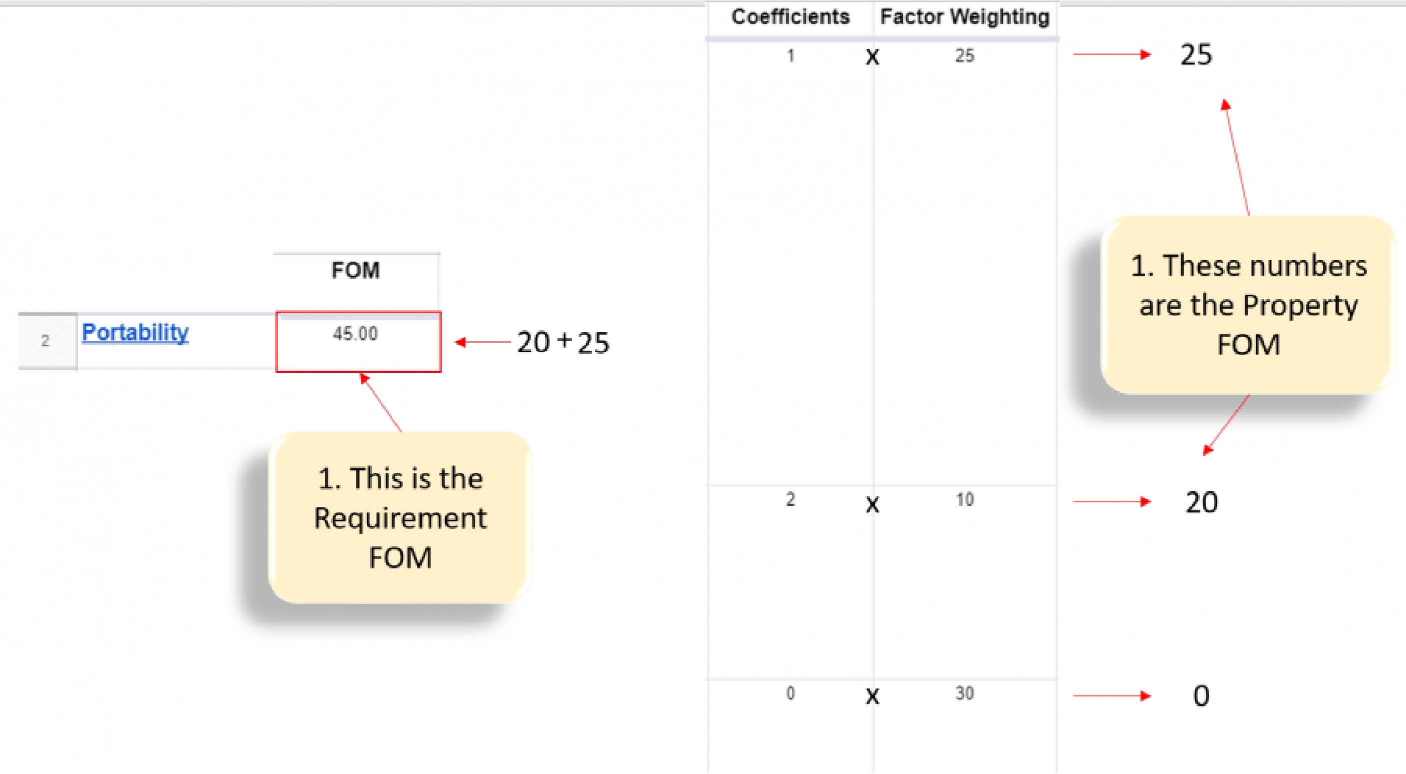


Coefficients	Factor Weighting
1	25
2	10
0	30

1. This column has data from the Coefficient sheet

2. This column has data from the Factor Weighting sheet

This data is then used to create the FOM(Figures of Merit). The formula to calculate the FOM = (C1)(FW1) + (C2)(FW2)+ (Cn)(FWn) for each property. These Property FOMs are then added into a Requirement FOM.



NOTE: Comparing Requirement FOMs from one vendor to another will cause a skewed comparison. To compare between vendors you must break the Requirement FOMs into its components. The components of Requirement FOM are the Property FOMs.

The Stakeholder's Award Team

Return to Top

The stakeholder's receive the Assessment Reports as well, if they deem the results satisfactory they will choose the product just reviewed.

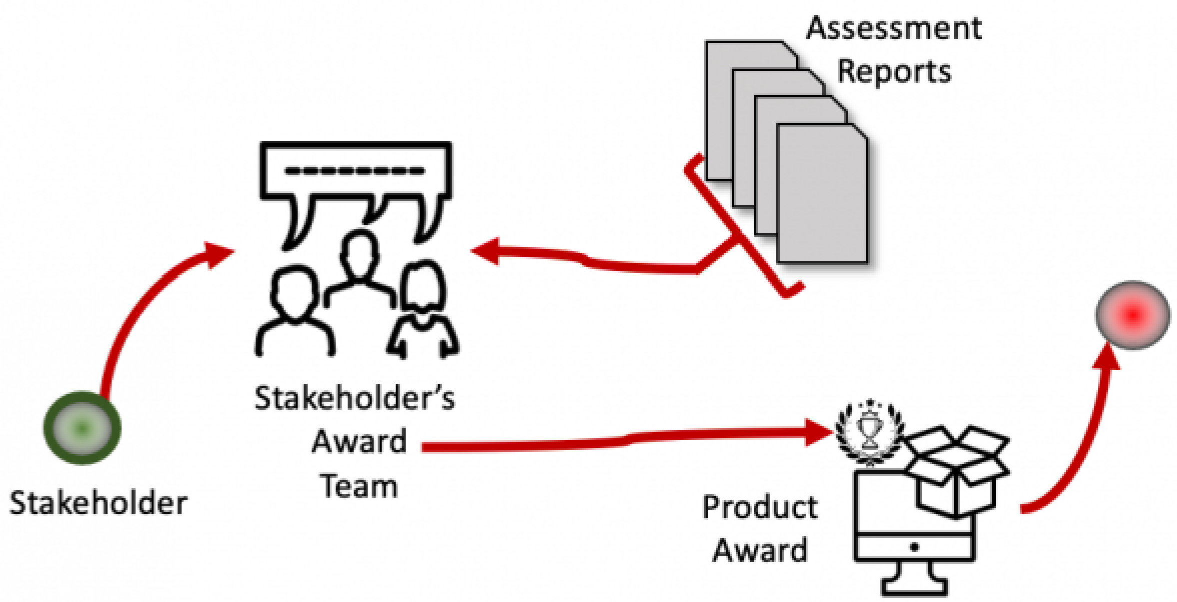


Figure 63: The award is usually your software is chosen to be used.

Appendices

Return to Reference Architecture (RA)

- Appendix A: Glossary of Terms Related to DIDO
- Appendix B: Standards Organizations
- Appendix C: Hardware Architectures
- Appendix D: Operating Systems
- Appendix E: Tools
- Appendix F: DDS Quality Of Service
- Appendix G: Tests
- Appendix H: Acronyms
- Appendix I: Cognitive Model
- Appendix J: Governance Model

Appendix A: Glossary of Terms Related to DIDO

Return to Reference Architecture (RA) or Return to Appendices

The following terms are used by DIDO in various documents, web pages, etc.

0-9	!-*	A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z		

!-*

Return to Glossary

Create a Glossary entry starting with '!-*' **Word or Expression →** Add page

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:0-9:start

Last update: **2021/03/12 16:03**

0-9

Return to Glossary

Create a Glossary entry starting with '0-9' **Word or Expression** →	Add page

- 16-Bit
- 32-Bit
- 64-Bit

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:00-9:start

Last update: **2021/06/01 22:10**

16-Bit

Return to Glossary

16-Bit Refers to the number of bits that can be processed or transmitted in parallel, or the number of bits used for single element in a data format. The term is often applied to the following:

- microprocessor: indicates the width of the registers. A 16-bit microprocessor can process data and memory addresses that are represented by 16 bits.\
- bus : indicates the number of wires in the bus. A 16-bit bus transmits 16 bits in parallel.
- graphics device, such as a scanner or digital camera : specifies the number of bits used to represent each pixel.
- operating system: refers primarily to the number of bits used to represent memory addresses. Windows 3.x is a 16-bit operating system, whereas Windows 95 and Windows NT are 32-Bit operating systems.
- expansion board: refers to how much data can be sent to and from the card in parallel. 8-bit cards are sometimes called half-size cards whereas 16-bit cards are referred to as full-size cards.

Source: 16-Bit

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:00-9:16bit

Last update: **2021/06/11 18:54**

32-Bit

Return to Glossary

32-Bit refers to the number of bits that can be transmitted or processed in parallel. In other words, 32-bits the number of bits that compose a data element.

- For a data bus, 32-bit means the number of pathways available, meaning that it has 32 pathways in parallel for data to travel.
- For microprocessors, it indicates the width of the registers and it can process any data and use memory addresses that are represented in 32-bits. This is part of the processor's architecture.
- For operating systems, 32-bits refer to how it handles data. It is used to represent a memory address and works in conjunction with the microprocessor.
- As for graphic devices like digital cameras or scanners, it refers to the number of bits used to represent the pixels. 24-bits are used for color information and 8-bits are used for the control information (alpha channel).

Source: 32-Bit

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:00-9:32bit

Last update: **2021/06/11 18:54**

64-Bit

Return to Glossary

64-Bit refers to the number of bits that can be processed or transmitted in parallel, or the number of bits used for single elements in data formats. It also refers to word sizes that define a certain class of computer architecture, buses, memory and Central Processing Unit (CPU).

Source: 64-Bit

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:00-9:64bit

Last update: **2021/06/11 18:55**

A

Return to Glossary

Create a Glossary entry starting with 'A' **Word or Expression →**	□ Add page

- Acceptance Testing
- Access Control
- Access Control Function
- Access Control List (ACL)
- Accessibility
- Accountability
- Adaptability
- Address Resolution Protocol (ARP)
- Address Resolution Protocol (ARP) Spoofing
- After Action Review (AAR)
- Aggregation Layer
- Agile Model
- American National Standards Institute (ANSI)
- American Standard for Information Interchange (ASCII)
- Analysability
- Application
- Application Container
- Application Layer
- Application Performance
- Application Programming Interface (API)
- Application Security
- Application Specific Integrated Circuit (ASIC)
- Appropriateness Recognizability
- Architecture Adaptability
- Argument
- Assurance
- Authentication
- Authenticity
- Authorization
- Automation Pyramid
- Availability

Acceptance Testing

Return to Glossary

Acceptance Testing is a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

There are various forms of acceptance testing:

- User acceptance Testing
- Business acceptance Testing
- Alpha Testing
- Beta Testing

Source: https://www.tutorialspoint.com/software_testing_dictionary/acceptance_testing.htm

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:acceptancetesting

Last update: **2020/12/20 21:01**

Access Control

Return to Glossary

Access Control is a security aspect that handles how user as well as system communicates and use resources. In order to enforce security, each and every access to the system and its resources should be controlled and should ensure only authorized access are allowed. This feature is mainly used to protect against unauthorized disclosure, corruption, modification, and destruction. It generally acts as the first line of defense to avoid the unauthorized access and entry. It comprises a set of controls that restrict access to resources based on the group membership, identity, clearance, physical & logical location and need-to-know. In addition, the access can be in the method of permission to consume, enter, control, restrict, use and protect the resource to guarantee three basic principles such as Availability, Confidentiality, and Integrity.

Source: https://www.hack2secure.com/blogs/an-introduction-to-core-security-concepts-cia-triad-and-aaa

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:accesscontrol

Last update: **2020/11/16 12:15**

Access Control Function

Return to Glossary

The **Access Control Function** prevents unauthorized interactions with an object. It includes both an access control decision function and an access control enforcement function. Within the context of Access Control, objects fulfill the roles of either target or initiator. The function requires access control information about the target, the initiator and the interaction. The initiator requests an interaction with the target from the access control function. The action control decision function decides whether access is permitted or denied on the basis of the access control information and the decision is enforced by the access control enforcement function.

Source: https://www.itu.int/itudoc/itu-t/com17/activity/def004_ww9.doc

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:accesscontrolfunction

Last update: **2020/11/16 12:16**

Access Control List (ACL)

Return to Glossary

An **Access Control List (ACL)** contains rules that grant or deny access to certain digital environments. There are two types of ACLs:

- Filesystem ACLs - filter access to files and/or directories. Filesystem ACLs tell operating systems which users can access the system, and what privileges the users are allowed.
- Networking ACLs - filter access to the network. Networking ACLs tell routers and switches which type of traffic can access the network, and which activity is allowed.

Source: https://www.imperva.com/learn/data-security/access-control-list-acl/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:acl

Last update: **2020/11/13 02:52**

Accessibility

Return to Glossary

Accessibility is degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:accessibility

Last update: **2021/06/14 21:06**

Accountability

Return to Glossary

See 4.2.4.5 Accountability

Accountability is the degree to which the actions of an entity can be traced uniquely to the entity.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&start=6

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:accountability

Last update: **2020/11/16 13:43**

Adaptability

Return to Glossary

See 4.2.1.1 Adaptability

Adaptability is the extent to which a software system adapts to change in its environment. An adaptable software system can tolerate changes in its environment without external intervention. For example, a dual-mode cell phone can find out by itself if any one of the two wireless standards it supports is available at its current location and if so it starts using that standard.

Source: Metrics for Software Adaptability

Address Resolution Protocol (ARP)

Return to Glossary

Address Resolution Protocol (ARP) is a procedure for mapping a dynamic Internet Protocol address (IP address) to a permanent physical machine address in a Local Area Network (LAN). The physical machine address is also known as a Media Access Control (MAC).

Source: Juniper Address Resolution Protocol

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:arp

Last update: **2021/06/14 21:22**

Address Resolution Protocol (ARP) Spoofing

Return to Glossary

Address Resolution Protocol (ARP) Spoofing is one way to initiate man-in-the-middle attacks. The attacker sends an Address Resolution Protocol (ARP) packet that spoofs the MAC address of another device on the Local Area Network (LAN). Instead of the switching device sending traffic to the proper Network Device, it sends the traffic to the device with the spoofed address that is impersonating the proper device. If the impersonating device is the attacker's machine, the attacker receives all the traffic from the switch that must have gone to another device. The result is that traffic from the switching device is misdirected and cannot reach its proper destination.

Source: Juniper ARP Spoofing

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:arpspoof

Last update: **2021/06/14 21:22**

After Action Review (AAR)

Return to Glossary

An **After Action Review (AAR)** is a structured review or de-brief (debriefing) process for analyzing what happened, why it happened, and how it can be done better by the participants and those responsible for the project or event. After-action reviews in the formal sense were originally developed by the U.S. Army. Formal AARs are used by all US military services and by many other non-US organizations. Their use has extended to business as a knowledge management tool and a way to build a culture of Accountability

Source: https://en.wikipedia.org/wiki/After-action_review

Aggregation Layer

Return to Glossary

Aggregation Layer, in Decentralized Finance (DeFi), consists of aggregators who connect various applications from the previous layer to provide a service to investors. For example, they might enable the seamless transfer of money between different financial instruments to maximize returns. In a physical setup, such trading actions would entail considerable paperwork and coordination. But a technology-based framework should smoothen the investing rails, allowing traders to switch between different services quickly. Lending and borrowing is an example of a service that exists on the aggregation layer. Banking services and crypto wallets are other examples.[9)]

Source: https://www.investopedia.com/decentralized-finance-defi-5113835

[9)] Rakesh Sharma, Investopedia, 24 March 2021, Decentralized Finance (DeFi) Definition, Accessed 24 May 2021, https://www.investopedia.com/decentralized-finance-defi-5113835

Agile Model

Return to Glossary

Agile Model is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a “big bang” launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

Source: Agile Model

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:agile

Last update: **2020/11/14 20:04**

American National Standards Institute (ANSI)

Return to Glossary

The **American National Standards Institute (ANSI)** oversees the creation and dissemination of various standards and measures, including business norms and standards in the United States.

The ANSI is a private, nonprofit organization and does not develop standards itself. Rather, it oversees the creation of voluntary standards for a variety of manufacturing processes, products, systems, services, and personnel in nearly every U.S. business sector. It also works to ensure that U.S. standards are consistent with international standards, enabling U.S. products to be sold and used abroad.

Source: https://www.investopedia.com/terms/a/ansi.asp

American Standard for Information Interchange (ASCII)

Return to Glossary

American Standard for Information Interchange (ASCII) is a method of encoding characters that is based on the order of alphabetic characters in the English language.

ASCII integer representations have printable and non-printable subsets. Printable characters are normal characters, and non-printable characters are characters used to represent keyboard keys, e.g., backspace, delete, and return.

Source: https://www.techopedia.com/definition/24322/american-standard-for-information-interchange-ascii

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:ascii

Last update: **2020/11/16 18:38**

Analysability

Return to Glossary

See 4.2.3.3 Analysability

Analysability is the capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:analysability

Last update: **2020/11/14 20:15**

Application

Return to Glossary

Application (more commonly known as an app) is software that bundles together certain features in a way that is accessible to a user. There are millions of apps on both the App Store and Android app stores, offering services (or verticals).

Source: Application

Application Container

Return to Glossary

An **Application Container** is a stand-alone, all-in-one package for a software application. Containers include the application binaries, plus the software dependencies and the hardware requirements needed to run, all wrapped up into an independent, self-contained unit.

Source: https://developer.hpe.com/blog/JM2009Z32ptYpRrm43Og/kubernetes-application-containers-managing-containers-and-cluster-resour

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:app_container

Last update: **2021/06/14 21:24**

Application Layer

Return to Glossary

1. In Open Systems Interconnection (OSI) Model, the **Application Layer** of the seven-layer Open Systems Interconnection (OSI) Model is the top layer that approaches protocols for application interaction with the network.

   With a focus on end-user services, the application layer helps to facilitate process-to-process connections over Internet protocol.

   Source: Application Layer

2. In Decentralized Finance (DeFi), the **Application Layer** as the name indicates, is where consumer-facing applications reside. These applications abstract underlying protocols into simple consumer-focused services. Most common applications in the cryptocurrency ecosystem, such as decentralized cryptocurrency exchanges and lending services, reside on this layer.[10)]

   Source: https://www.investopedia.com/decentralized-finance-defi-5113835

[10)] Rakesh Sharma, Investopedia, 24 March 2021, Decentralized Finance (DeFi) Definition, Accessed 24 May 2021, https://www.investopedia.com/decentralized-finance-defi-5113835

Application Performance

Return to Glossary

Application Performance is the measurement of the real-world performance and availability of applications. It is particularly used with remote and cloud computing applications being run in remote servers and served over a network such as the Internet. Application performance is a good indicator of the level of service that a provider is offering and is one of the top monitored IT metrics.

Source: Application Performance

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:appperform

Last update: **2020/11/16 21:09**

Application Programming Interface (API)

Return to Glossary

An **application programming interface (API)** is a set of protocols, routines, functions and/or commands that programmers use to develop software or facilitate interaction between distinct systems. APIs are available for both desktop and mobile use and are typically useful for programming Graphical User Interface (GUI) components, as well as allowing a software program to request and accommodate services from another program.

Source: Application Programming Interface (API)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:api

Last update: **2020/11/13 02:51**

Application Security

Return to Glossary

Application Security is the process of developing, adding, and testing security features within applications to prevent security vulnerabilities against threats such as unauthorized access and modification.

Application Security describes security measures at the application level that aim to prevent data or code within the app from being stolen or hijacked. It encompasses the security considerations that happen during application development and design, but it also involves systems and approaches to protect apps after they get deployed.

Application Security may include hardware, software, and procedures that identify or minimize security vulnerabilities. A router that prevents anyone from viewing a computer's Internet Protocol address (IP address) from the Internet is a form of hardware Application Security. But security measures at the application level are also typically built into the software, such as an application firewall that strictly defines what activities are allowed and prohibited. Procedures can entail things like an Application Security routine that includes protocols such as regular testing.

Source: https://www.vmware.com/topics/glossary/content/application-security

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:applicationsecurity

Last update: **2020/11/15 17:48**

Application Specific Integrated Circuit (ASIC)

Return to Glossary

Application Specific Integrated Circuit (ASIC) is basically an integrated circuit that is designed specifically for an individual purpose or application. Strictly speaking, this also implies that an ASIC is built for one and only one customer. The opposite of an ASIC is a standard product or general-purposed IC, such as a logic gate or a general-purposed microcontroller chip, both of which can be used across a large range of electronic applications. ASICs are usually classified into one of three categories; full custom, semi-custom, and structured

Source: Application Specific Integrated Circuit (ASIC)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:asic

Last update: **2020/11/13 02:52**

Appropriateness Recognizability

Return to Glossary

Appropriateness Recognizability is degree to which users can recognize whether a product or system is appropriate for their needs.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:appropriateness_recognizability

Last update: **2021/06/14 21:24**

Architecture Adaptability

Return to Glossary

Architecture Adaptability is when the connectors between software components change without having to change the components. This again comes down to having well-defined, stable APIs for the connectors. Unix File System (UnixFS) is a connector between software components and the physical filesystem. For example, the UnixFS library can be exchanged for the InterPlanetary File System (IPFS) UnixFS connector and the software component should have no to minimal impact. [11].

Source: https://personal.utdallas.edu/~chung/ftp/sqm.pdf

[11] Nary Subramanian and Lawrence Chung, Metrics for Software Adaptability, University of Texas - Dallas, Accessed 29 July 2020, https://personal.utdallas.edu/~chung/ftp/sqm.pdf

Argument

Return to Glossary

An **Argument** represents logical propositions intended to support a Claim through reasoning or logic that links Evidence to a claim. Arguments define the relationships directly linking each claim and/or sub-claims, and piece of evidence, used by an argument to the claims immediately supported by the argument. An argument is the explanation of how the evidence can be interpreted as supporting a claim or sub-claim.

Arguments can also include any unusual events or conditions that are within the context of the claim. The argument can contain considerations of potential causes of failure and appropriate corrective actions if failure occurs. Hence, an argument may include conditions, assumptions, and judgments about the system, its use, and its operational environment, threats, and likelihood of occurrence, for which the claims and evidence are being marshaled as part of an overall assurance case.

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4548534/#b13-v115.n03.a05

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:argument

Last update: **2020/11/13 02:51**

Assurance

Return to Glossary

Assurance is the measure of confidence that the security features, practices, procedures, and architecture of an information system accurately mediates and enforces the security policy. - CNSS 4009 IA Glossary

Source: Committee on National Security Systems, CNS 4009 Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:assurance

Last update: **2020/11/13 02:59**

Authentication

Return to Glossary

Authentication is the process of recognizing a user’s identity. It is the mechanism of associating an incoming request with a set of identifying credentials. The credentials provided are compared to those on a file in a database of the authorized user’s information on a local operating system or within an authentication server.

Source: Authentication

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:authentication

Last update: **2020/11/15 17:44**

Authenticity

Return to Glossary

See 2.2.4.4 Authenticity

Authenticity is the degree to which the identity of a subject or resource can be proved to be the one claimed.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&start=6

Authorization

Return to Glossary

Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features. Authorization is normally preceded by authentication for user identity verification. System administrators (SA) are typically assigned permission levels covering all system and user resources.

During authorization, a system verifies an authenticated user's access rules and either grants or refuses resource access.

Source: Authorization

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:authorization

Last update: **2020/11/13 02:52**

Automation Pyramid

Return to Glossary

The **Automation Pyramid** classifies the different IT layers of industrial automated production plants. Every layer or level has is own tasks and IT infrastructure within the production plants. In general, the automation pyramid is divided into 6 layers.

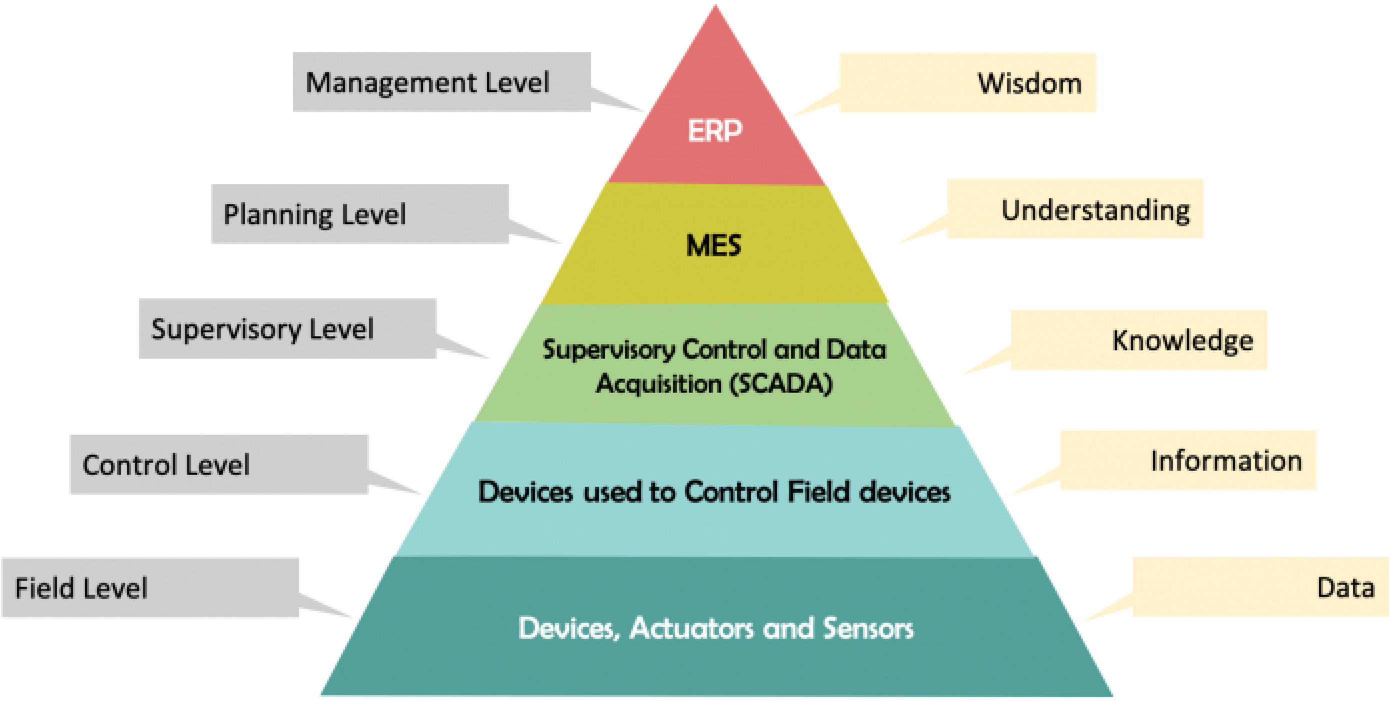


Figure 3: Automation Pyramid

Source: Automation Pyramid

Availability

Return to Glossary

See 4.2.2.2 Availability

Availability is the probability that a system will work as required when required during the period of a mission. The mission could be the 18-hour span of an aircraft flight. The mission period could also be the 3 to 15-month span of a military deployment. Availability includes non-operational periods associated with reliability, maintenance, and logistics.

Source: Availability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:a:availability

Last update: **2020/11/13 02:52**

B

Return to Glossary

Create a Glossary entry starting with 'B' **Word or Expression →**	Add page

- Backus–Naur Form (BNF)
- Bandwidth
- Big-Endian
- BigQuery
- Bill of Lading (BL or BoL)
- Biometrics
- Bit Error
- Bitcoin Wallet
- Black Box Testing
- Block Producers
- Block Validators
- Blockchain
- Blockchain Network
- Bluetooth
- Bootstrap
- Bridge
- Brownfield
- Bylaws
- Byzantine Fault Tolerance
- Byzantine Generals Problem

Backus–Naur Form (BNF)

Return to Glossary

Backus–Naur Form or **Backus Normal Form (BNF)** is a metasyntax notation for context-free grammars, often used to describe the syntax of languages used in computing, such as computer programming languages, document formats, instruction sets and communication protocols. They are applied wherever exact descriptions of languages are needed: for instance, in official language specifications, in manuals, and in textbooks on programming language theory.

Many extensions and variants of the original Backus–Naur notation are used; some are exactly defined, including extended Backus–Naur form (EBNF) and augmented Backus–Naur form (ABNF).

Source: https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bnf

Last update: **2021/06/14 21:27**

Bandwidth

Return to Glossary

Bandwidth is a broad term defined as the bit-rate measure of the transmission capacity over a network communication system. Bandwidth is also described as the carrying capacity of a channel or the data transfer speed of that channel. However, broadly defined, bandwidth is the capacity of a network. Bandwidth exists in physical or wireless communication networks.

Source: https://www.techopedia.com/definition/5245/bandwidth

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bandwidth

Last update: **2021/06/14 21:27**

Big-Endian

Return to Glossary

Big-Endian refers to the way that data is sequentially stored in computer memory. Just as in books or magazines, where the first word appears in the top-left-hand corner of each page, the data in a big-endian system is organized such that the most significant digits or bytes appear in the upper left corner of a memory page, while the least significant ones appear in the bottom right-hand corner.

Note: This is in contrast to Little-Endian systems.

Source: Big-Endian

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bigendian

Last update: **2021/06/14 21:28**

Bill of Lading (BL or BoL)

Return to Glossary

A **Bill of Lading (BL or BOL)** is a legal document issued by a carrier to a shipper that details the type, quantity, and destination of the goods being carried. A bill of lading also serves as a shipment receipt when the carrier delivers the goods at a predetermined destination. This document must accompany the shipped products, no matter the form of transportation, and must be signed by an authorized representative from the carrier, shipper and receiver.

Source: https://www.investopedia.com/terms/b/billoflading.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bol

Last update: **2021/06/14 21:28**

Biometrics

Return to Glossary

Biometrics are a group of digital security methods that rely on biological or physiological attributes, used to prevent data breaches such as credit card hacks or unauthorized log-ins. Biometrics uses criteria that is physically unique to an individual that can prove their identity, such as a fingerprint or voice pattern, rather than relying on passwords or PIN codes that can be more easily hacked or stolen.

Source: https://www.investopedia.com/terms/b/biometrics.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:biometrics

Last update: **2020/11/15 20:00**

Bitcoin Wallet

Return to Glossary

A **Bitcoin wallet** is a software program where Bitcoins are stored. To be technically accurate, Bitcoins are not stored anywhere; there is a private key (secret number) for every Bitcoin address that is saved in the Bitcoin wallet of the person who owns the balance. Bitcoin wallets facilitate sending and receiving Bitcoins, and give ownership of the Bitcoin balance to the user. The Bitcoin wallet comes in many forms; desktop, mobile, web, and hardware are the four main types of wallets.

Source: Bitcoin Wallet

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bitcoin_wallet

Last update: **2020/11/13 02:59**

Bit Error

Return to Glossary

Bit Error occurs when a bit on a data stream is altered due to noise, interference, distortion or bit synchronization errors.

Source: Local

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:biterr

Last update: **2021/06/14 21:28**

Black Box Testing

Return to Glossary

Black Box Testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

Black box testing can be done in following ways:

1. Syntax Driven Testing – This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers,language that can be represented by context free grammar. In this, the test cases are generated so that each grammar rule is used at least once.

1. Equivalence partitioning – It is often seen that many type of inputs work similarly so instead of giving all of them separately we can group them together and test only one input of each group. The idea is to partition the input domain of the system into a number of equivalence classes such that each member of class works in a similar way, i.e., if a test case in one class results in some error, other members of class would also result into same error.

Source: https://www.geeksforgeeks.org/software-engineering-black-box-testing/

Blockchain

Return to Glossary

Blockchain is a critical part of the bitcoin peer-to-peer payment system. The bitcoin system works using a blockchain ledger to record transactions. Bitcoin is a global cryptocurrency that can be used as a medium of exchange. However, while many parties have started to accept bitcoin as a currency, it is still controversial and poses risks in terms of security and stability.

Source: Blockchain

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:blkchn

Last update: **2020/11/13 02:59**

Blockchain Network

Return to Glossary

A **Blockchain_network** is a technical infrastructure that provides ledger and smart contract (chaincode) services to applications. Primarily, smart contracts are used to generate transactions which are subsequently distributed to every peer node in the network where they are immutably recorded on their copy of the ledger. The users of applications might be end users using client applications or blockchain network administrators.

Source: Blockchain Network

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:blockchain_network

Last update: **2020/11/13 03:02**

Block Producers

Return to Glossary

Block Producers in Delegated Proof of Stake (DPoS) are full nodes responsible for creating and signing new blocks. They are limited in number, and are elected by the voters.

Source: Block Producers

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:block_producers

Last update: **2020/11/13 02:59**

Block Validators

Return to Glossary

Block Validators in Delegated Proof of Stake (DPoS) refer to full nodes that verify the blocks created by Block Producers and follow consensus rules. Any user is able to run a block validator and verify the network. (This can be confusing, since in Casper's PoS, the word "validators" refers to those who create blocks).

Source: Block Validators

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:block_validators

Last update: **2020/11/13 02:59**

Bluetooth

Return to Glossary

Bluetooth is an open wireless technology standard for transmitting fixed and mobile electronic device data over short distances. Bluetooth was introduced in 1994 as a wireless substitute for RS-232 cables.

Bluetooth communicates with a variety of electronic devices and creates personal networks operating within the unlicensed 2.4 GHz band. Operating range is based on device class. A variety of digital devices use Bluetooth, including MP3 players, mobile and peripheral devices and personal computers.

Source: Bluetooth

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bluetooth

Last update: **2020/11/14 17:41**

Bootstrap

Return to Glossary

Bootstrap is a program that initializes the Operating System (OS) during startup. The term bootstrap or bootstrapping originated in the early 1950s. It referred to a bootstrap load button that was used to initiate a hardwired bootstrap program, or smaller program that executed a larger program such as the OS. The term was said to be derived from the expression “pulling yourself up by your own bootstraps,” starting small and loading programs one at a time while each program is “laced” or connected to the next program to be executed in sequence.

Source: Technopedia-Bootstrap

Bridge

Return to Glossary

A **Bridge** connect two or more hosts or network segments together. The basic role of bridges in network architecture is storing and forwarding frames between the different segments that the bridge connects. They use hardware Media Access Control (MAC) addresses for transferring frames. By looking at the MAC address of the devices connected to each segment, bridges can forward the data or block it from crossing. Bridges can also be used to connect two physical LANs into a larger logical LAN.

Bridges work only at the Physical and Data Link layers of the Open Systems Interconnection (OSI) Model. Bridges are used to divide larger networks into smaller sections by sitting between two physical network segments and managing the flow of data between the two.

Bridges are like hubs in many respects, including the fact that they connect LAN components with identical protocols. However, bridges filter incoming data packets, known as frames, for addresses before they are forwarded. As it filters the data packets, the bridge makes no modifications to the format or content of the incoming data. The bridge filters and forwards frames on the network with the help of a dynamic bridge table. The bridge table, which is initially empty, maintains the LAN addresses for each computer in the LAN and the addresses of each bridge interface that connects the LAN to other LANs. Bridges, like hubs, can be either simple or multiple port.

Bridges have mostly fallen out of favor in recent years and have been replaced by switches, which offer more functionality. In fact, switches are sometimes referred to as “multiport bridges” because of how they operate.

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bridge

Last update: **2020/11/14 15:54**

Brownfield

Return to Glossary

Brownfield refers to the implementation of new systems to resolve IT problem areas while accounting for established systems. New software architecture must account for existing and running software.

A commonly used IT term, Brownfield was borrowed from the building industry, where brownfield land describes a geographical location where new buildings may be constructed after considering the area's established structures and services.

Source: Brownfield

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:brownfield

Last update: **2020/11/14 20:07**

Bylaws

Return to Glossary

Bylaws are legal documents that establish the internal structure and governing rules of the organization. Bylaws provide the framework for internal governance and day-to-day operations of its governing structure.

Source: Bylaws

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:b:bylaw

Last update: **2020/11/13 03:02**

Byzantine Fault Tolerance

Return to Glossary

Byzantine Fault Tolerance is the characteristic of a system that is able to tolerate a class of failures called the Byzantine Generals’ Problem[12].

[12] “Understanding Blockchain Fundamentals, Part 1: Byzantine Fault Tolerance”, Georgios Konstantopoulos, 1 December 2017, Byzantine Fault Tolerance

Byzantine Generals Problem

Return to Glossary

Byzantine Generals Problem is when a reliable computer system must be able to cope with the failure of one or more of its components. A failed component may exhibit a type of behavior that is often overlooked – namely, sending conflicting information to different parts of the system. The problem of coping with this type of failure is expressed abstractly as the Byzantine Generals Problem. [13]

[13] The Byzantine Generals Problem“, Leslie Lamport, Robert Shostak, and Marshal Pesse, SRI International, July 1983, Byzantine Generals Problem

C

Return to Glossary

Create a Glossary entry starting with 'C' **Word or Expression** →		Add page

- Cable Subscriber Protection
- California Consumer Privacy Act (CCPA)
- Capability Maturity Model Integration (CMMI)
- Category 5 (Cat-5)
- Category 6 (Cat-6)
- Category 7 (Cat-7)
- Category 8 (Cat-8)
- Central Processing Unit (CPU)
- Charter
- Children's Online Privacy Protection Act (COPPA)
- Claim
- Class
- Client
- Client-Server
- Cloud Elasticity
- Coins
- Comma Separated Values (CSV)
- Command Line Interface (CLI)
- Command Shell
- Commercial Off-The-Shelf (COTS)
- Common Intermediate Language (CIL)
- Common Language Runtime (CLR)
- Common Object Request Broker Architecture (CORBA)
- Communication Protocol
- Communications Model
- Community of Interest (CoI)
- Compiler
- Complex Instruction Set Computer (CISC)
- Computer Architecture
- Computing Platform
- Conceptual Schema
- Condition
- Confidentiality
- Confidentiality Agreement
- Configuration Management (CM)
- Conformance Specification
- Consensus Algorithm
- Container

- Container Engine
- Container Host
- Container OS
- Control Level
- Copyleft
- Copyright
- Cryptocurrency
- CyberSecurity Culture (CSC)

Cable Subscriber Protection

Return to Glossary

Cable Subscriber Protection 47 U.S. Code § 551 provides access to all Personal Identifiable Information (PII) regarding that subscriber which is collected and maintained by a cable operator. Such information shall be made available to the subscriber at reasonable times and at a convenient place designated by such cable operator. A cable subscriber shall be provided reasonable opportunity to correct any error in such information.

Source: https://www.law.cornell.edu/uscode/text/47/551

California Consumer Privacy Act (CCPA)

Return to Glossary

California Consumer Privacy Act (CCPA) gives consumers more control over the personal information that businesses collect about them and the CCPA regulations provide guidance on how to implement the law. This landmark law secures new privacy rights for California consumers, including:

- The right to know about the personal information a business collects about them and how it is used and shared;
- The right to delete personal information collected from them (with some exceptions);
- The right to opt-out of the sale of their personal information; and
- The right to non-discrimination for exercising their CCPA rights.

Businesses are required to give consumers certain notices explaining their privacy practices. The CCPA applies to many businesses, including data brokers.

Source: https://oag.ca.gov/privacy/ccpa

Capability Maturity Model Integration (CMMI)

Return to Glossary

Capability Maturity Model Integration (CMMI®) is a capability improvement model that can be adapted to solve any performance issue at any level of the organization in any industry. The Model provides guidelines and recommendations for helping your organization diagnose problems and improve performance. Used by over 10,000 organizations from more than 106 countries all over the world, CMMI helps you to identify and achieve measurable business goals.

Product and service providers, procurement officers and human capital managers all face different challenges. Because CMMI provides guidance for improvement across multiple process disciplines in your organization, you can use it to identify and achieve any measurable performance goal.

Source:
https://cmmiinstitute.zendesk.com/hc/en-us/articles/216947067-What-is-CMMI-What-is-the-CMMI-Model-

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cmmi

Last update: **2020/11/14 12:40**

Category 5 (Cat-5)

Return to Glossary

Category 5 (Cat-5) is network cabling that consists of four twisted pairs of copper wire terminated by RJ45 connectors. Cat-5 cabling supports frequencies up to 100 MHz and speeds up to 1000 Mbps. It can be used for ATM, token ring, 1000Base-T, 100Base-T, and 10Base-Tnetworking.

Computers hooked up to LANs are connected using Cat-5 cables, so if you're on a LAN, most likely the cable running out of the back of your PC is Category 5.

Cat-5 is based on the EIA/TIA 568 Commercial Building Telecommunications Wiring Standard developed by the Electronics Industries Association as requested by the Computer Communications Industry Association in 1985.

Source: Category 5 (Cat-5)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cat5

Last update: **2020/11/16 18:31**

Category 6 (Cat-6)

Return to Glossary

Category 6 (Cat-6) network cabling is used as the cabling infrastructure for 10BASE-T (Ethernet), 100BASE-TX (Fast Ethernet), 1000BASE-T (Gigabit Ethernet, or GbE) and 10GBASE-T (10-Gigabit Ethernet, or 10 GbE) networks. The Cat 6 standard provides performance of up to 250 MHz (500 MHz for the newer Cat 6a standard) and can be used up to a maximum length of 100 meters (55 meters for 10GBASE-T networks).

The Cat 6 standard was first released in 2002 as part of the Telecommunications Industry Association’s TIA/EIA-568-B.2-1 document specification. Cat 6 is backward compatible with the Cat 3, Cat 5 and Cat 5e cable standards, and as with Cat 5 and Cat 5e cabling, Cat 6 cables consist of four unshielded twisted pairs (UTP) of copper wire terminated by RJ45 connectors.

In addition to its support for higher performance than the Cat 5 specification, the Cat 6 standard also includes more stringent specifications for crosstalk and system noise. While Cat 6 is expected to supersede both Cat 5 and Cat 5e cabling in the future, all three types of cables continue to be popular for use in network installations.

Source: https://www.webopedia.com/TERM/C/cat_6.html

Category 7 (Cat-7)

Return to Glossary

Category 7 (Cat-7) network cabling is used as a cabling infrastructure for 1000BASE-T (Gigabit Ethernet, or GbE) and 10GBASE-T (10-Gigabit Ethernet, or 10 GbE) networks. The Cat 7 standard provides performance of up to 600 MHz (1000 MHz for the Cat-7a, or Augmented Category 7 standard) and can be used up to a maximum length of 100 meters.

Category 7 cable is able to achieve higher performance than preceding Ethernet standards such as Cat 5, Cat 5e and Cat 6 by requiring each of its twisted wire pairs to be fully shielded. This is known as Screen Shielded Twisted Pair (SSTP) or Screened Foiled Twisted Pair (SFTP) wiring, and it almost completely eliminates alien crosstalk while significantly improving noise resistance.

The Cat 7 standard was published in 2002 by the International Organization for Standardization (ISO) and is also known as Class F cabling. While more expensive than Cat 5e and Cat 6 cabling, Cat-7 cabling does have a 15-year lifecycle (compared to estimated 10-year lifecycles for Cat 5e and Cat 6), which helps improve its overall

Source: Category 7 (Cat-7)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cat7

Last update: **2020/11/16 18:35**

Category 8 (Cat-8)

Return to Glossary

Category 8 (Cat-8) Ethernet cables different than previous versions because they support a frequency of 2 GHz. Cat-8 is limited to a 30-meter 2-connector channel and also require shielding around the wire structures. Cat-8 can support speeds of 25 Gbps or in some cases 40 Gbps. The looks and physical appearance of cat8 are similar to the previous cables and can be terminated with RJ-45 or non-RJ-45 connections.

A major difference between Cat-8 and Category 7 (Cat-7) cables are the frequencies: Cat-7 is 600 MHz speed and cat-8 is about 2000 MHz speed.

Source: Category 8 (Cat-8)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cat8

Last update: **2020/11/16 18:36**

Central Processing Unit (CPU)

Return to Glossary

Central Processing Unit (CPU) is the primary component of a computer that processes instructions. It runs the operating system and applications, constantly receiving input from the user or active software programs. It processes the data and produces output, which may stored by an application or displayed on the screen.

Source: Central Processing Unit (CPU)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cpu

Last update: **2020/11/13 03:00**

Charter

Return to Glossary

A **Charter** is a legal document that creates for-profit or nonprofit organizations. Frequently called articles of incorporation, a charter brings the organization into existence as a legal entity. Charters must be filed with an approval authority such as the secretary of state for the state where the organization is located.

Source: Charter

Children's Online Privacy Protection Act (COPPA)

Return to Glossary

The **Children's Online Privacy Protection Act (COPPA) (15 U.S. Code § 6501)** protects children's information at the federal level, which prohibits the collection of any information from a child under the age of 13 online and from digitally connected devices, and requires publication of privacy notices and collection of verifiable parental consent when information from children is being collected.

Source: https://iclg.com/practice-areas/data-protection-laws-and-regulations/usa

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:coppa

Last update: **2021/06/14 21:30**

Claim

Return to Glossary

A **Claim** is a statement asserting some characteristic, property, or behavior of the software or system that can be evaluated for truthfulness, is demonstrable, and is supported by arguments based on objective evidence. A claim may be further decomposed into Sub-Claim, and expressed either as a positive or negative statement.

For example, one can declare positive claims about the requirements-based, quality properties of software, such as its dependability or availability, or one can make negative claims about the same software by claiming that the code does not contain specific weaknesses and vulnerabilities in the design and implementation that could be exploited to break or compromise the system.

See also:

- Sub-Claim
- Argument
- Evidence

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4548534/#b13-v115.n03.a05

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:claim

Last update: **2020/11/13 02:59**

Class

Return to Glossary

A **Class**, in Programming Lanuages, are a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

For Example: Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, Car is the class, and wheels, speed limits, mileage are their properties.

A Class is a user-defined data type that has data members and member functions. Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behavior of the objects in a Class.

In the above example of class Car, the data member will be speed limit, mileage, etc. and member functions can apply brakes, increase speed, etc. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Source: https://www.geeksforgeeks.org/c-classes-and-objects/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:class

Last update: **2021/06/14 21:30**

Client

Return to Glossary

A **Client** is the receiving end of a service or the requestor of a service in a Client-Server model type of system. The client is most often located on another system or computer, which can be accessed via a network. This term was first used for devices that could not run their own programs, and were connected to remote computers that could via a network. These were called dumb terminals and they were served by time-sharing mainframe computers.

Source: https://www.techopedia.com/definition/437/client

Client-Server

Return to Glossary

The Client-Server model helped developers address the scalability issues of the Point-to-Point communication model. Client-server networks designate one special server node that connects simultaneously to many client nodes, as illustrated below. Client-server is a “many-to-one” communications architecture.

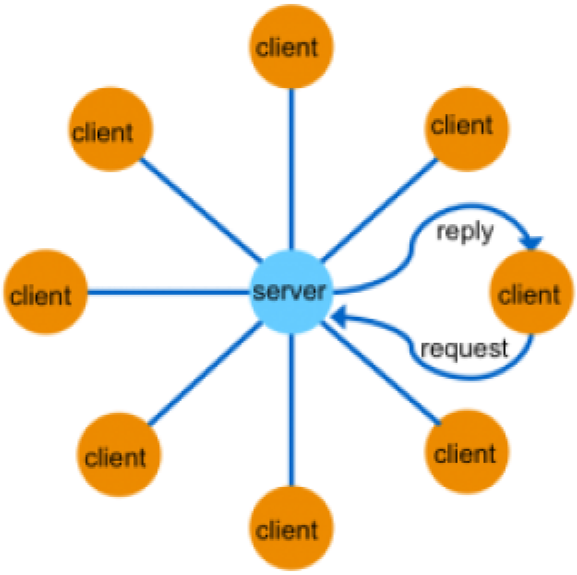


Figure 4: Client-Server Communication

The client-server network architecture works best when information is centralized, such as in databases, transaction processing systems, and file servers. However, if information is being generated at multiple nodes, a client-server architecture requires that all information are sent to the server for later redistribution to the clients. This approach is inefficient and precludes deterministic communications, since the client does not know when new information is available. The time between when the information is available on the server, and when the client asks and receives it adds a variable latency to the system. Example of Client-Server Communication

Ordering pizza over the phone is an example of client-server communication. Clients must know the phone number of the pizza parlor to place an order. The parlor can handle many orders without knowing ahead of time where people (clients) are located. After the order (request), the parlor asks the client where the response (pizza) should be sent. In the client-server model, each response is tied to a prior request. As a result, the response can be tailored to each request. In other words, each client makes a request (order) and each reply (pizza) is made for one specific client in mind.

Provided by: https://community.rti.com/glossary/client-server

Cloud Elasticity

Return to Glossary

See 4.2.5 Elasticity

Cloud Elasticity refers to the ability of a cloud service to provide on-demand offerings, nimbly switching resources when demand goes up or down. It is often an immediate reaction to clients dropping or adding services in real time.

Cloud elasticity is also known as rapid elasticity.

Source: https://www.techopedia.com/definition/30992/cloud-elasticity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cloudelasticity

Last update: **2020/11/22 20:51**

Coins

Return to Glossary

Coins (also often called *altcoins* or *alternative cryptocurrency coins*) are digital money, created using encryption techniques, that store value over time. Basically they are the digital equivalent of money. Bitcoin is the most famous example. Bitcoin is based on blockchain — public and distributed digital ledger — where all transactions can be seen. Data is stored collectively and shared between participants of a blockchain network. Blockchain guarantees transparency and reduces fraud.

Coins have the same characteristics as money: they are fungible, divisible, acceptable, portable, durable and have limited supply. Most ambitious crypto enthusiasts insist that coins will replace conventional money in the future.

The main characteristics of coins are:

1. they are tied to public-open blockchain—anyone is allowed to join and participate in the network;
2. they may be sent, received or mined.

Coins are not meant to perform any functions beyond acting as money.

Compare to tokens

Source: Coins

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:coins

Last update: **2020/11/13 03:00**

Comma Separated Values (CSV)

Return to Glossary

A **Comma Separated Values (CSV)** file contains different values separated by a delimiter, which acts as a database table or an intermediate form of a database table. In other words, a CSV file is a set of database rows and columns stored in a text file such that the rows are separated by a new line while the columns are separated by a semicolon or a comma. A CSV file is primarily used to transport data between two databases of different formats through a computer program.

Source: https://www.techopedia.com/definition/24364/comma-separated-values-file-csv

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:csv

Last update: **2021/06/14 21:31**

Command Line Interface (CLI)

Return to Glossary

Command Line Interface (CLI) is is a text-based interface that is used to operate software and operating systems while allowing the user to respond to visual prompts by typing single commands into the interface and receiving a reply in the same way.

CLI is quite different from the Graphical User Interface (GUI) that is presently being used in the latest operating systems.

Source: Command Line Interface (CLI)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cli

Last update: **2021/06/14 21:32**

Command Shell

Return to Glossary

The **Command Shell** is the command processor interface. The command processor is the program that executes operating system commands. The shell therefore, is the part of the command processor that accepts commands. After verifying that the commands are valid, the shell sends them to another part of the command processor to be executed. UNIX systems offer a choice between several different shells, the most popular being the Cshell, the Bourne shell, and the Korn shell. Each offers a somewhat different command language.

Source: https://www.webopedia.com/definitions/shell/

Commercial Off-The-Shelf (COTS)

Return to Glossary

Commercial Off-The-Shelf (COTS) is an adjective that describes software or hardware products that are ready-made and available for sale to the general public. For example, Microsoft Office is a COTS product that is a packaged software solution for businesses. COTS products are designed to be implemented easily into existing systems without the need for customization.

Source: Commercial Off-The-Shelf (COTS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cots

Last update: **2021/06/14 21:33**

Common Intermediate Language (CIL)

Return to Glossary

Common Intermediate Language (CIL), formerly called Microsoft Intermediate Language (MSIL) or Intermediate Language (IL),[1] is the intermediate language binary instruction set defined within the Common Language Infrastructure (CLI) specification.[2] CIL instructions are executed by a CLI-compatible runtime environment such as the Common Language Runtime. Languages which target the CLI compile to CIL. CIL is object-oriented, stack-based bytecode. Runtimes typically just-in-time compile CIL instructions into native code.

Source: Common Intermediate Language (CIL)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cil

Last update: **2020/11/13 12:30**

Common Language Runtime (CLR)

Return to Glossary

Common Language Runtime (CLR) is is a managed execution environment that is part of Microsoft's .NET framework. CLR manages the execution of programs written in different supported languages.

CLR transforms source code into a form of bytecode known as Common Intermediate Language (CIL). At run time, CLR handles the execution of the CIL code.

Source: Common Language Runtime (CLR)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:clr

Last update: **2020/11/13 13:03**

Common Object Request Broker Architecture (CORBA)

Return to Glossary

The **Common Object Request Broker Architecture (CORBA)** is a standard developed by the Object Management Group (OMG) to provide interoperability among distributed objects. CORBA is the world's leading middleware solution enabling the exchange of information, independent of hardware platforms, programming languages, and operating systems. CORBA is essentially a design specification for an Object Request Broker (ORB), where an ORB provides the mechanism required for distributed objects to communicate with one another, whether locally or on remote devices, written in different languages, or at different locations on a network.

Source: Common Object Request Broker Architecture (CORBA)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:corba

Last update: **2020/11/13 03:00**

Communication Protocol

Return to Glossary

Communication protocols are formal descriptions of digital message formats and rules. They are required to exchange messages in or between computing systems and are required in telecommunications. Communication protocols cover authentication, error detection and correction, and signaling. They can also describe the syntax, semantics, and synchronization of analog and digital communications.

Source: Communication Protocol

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:communication_protocol

Last update: **2020/11/13 13:02**

Communications Model

The **Communications Model** underlying the network middleware is the most important factor in how applications communicate. The communications model impacts the performance, the ease to accomplish different communication transactions, the nature of detecting errors, and the robustness to different error conditions. Unfortunately, there is no “one size fits all” approach to distributed applications. Different communications models are better suited to handle different classes of application domains.

We commonly discuss three main types of network communications models:

- Point-to-Point
- Client-Server
- Publish-Subscribe

Source: https://community.rti.com/glossary/communications-model

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:communications_model

Last update: **2021/06/11 16:38**

Community of Interest (CoI)

Return to Glossary

Community of Interest (CoI) is a collaborative group of users who exchange information in pursuit of their shared goals, interests, missions, or business processes, and who therefore must have a shared vocabulary for the information they exchange. The group exchanges information within and between systems to include security domains.

Source: Community of Interest

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:coi

Last update: **2020/11/13 22:02**

Compiler

Return to Glossary

A **Compiler** is a software program that transforms high-level source code that is written by a developer in a high-level programming language into a low level object code (binary code) in machine language, which can be understood by the processor. The process of converting high-level programming into machine language is known as compilation.

The processor executes object code, which indicates when binary high and low signals are required in the arithmetic logic unit of the processor.

Source: Compiler

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:compiler

Last update: **2020/11/13 13:03**

Complex Instruction Set Computer (CISC)

Return to Glossary

Complex Instruction Set Computer (CISC) is a computer in which single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions. The term was retroactively coined in contrast to Reduced Instruction Set Computer (RISC) and has therefore become something of an umbrella term for everything that is not RISC, from large and complex mainframe computers to simplistic Microcontroller where memory load and store operations are not separated from arithmetic instructions. A modern RISC processor can therefore be much more complex than, say, a modern microcontroller using a CISC-labeled instruction set, especially in the complexity of its electronic circuits, but also in the number of instructions or the complexity of their encoding patterns. The only typical differentiating characteristic is that most RISC designs use uniform instruction length for almost all instructions, and employ strictly separate load/store-instructions.

Source: Complex Instruction Set Computer (CISC)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cisc

Last update: **2021/06/14 21:33**

Computer Architecture

Return to Glossary

Computer Architecture is a science or a set of rules stating how computer software and hardware are joined together and interact to make a computer work. It not only determines how the computer works but also of which technologies the computer is capable.

Source: https://www.computersciencedegreehub.com/faq/what-is-computer-architecture/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:computer_architecture

Last update: **2021/06/15 08:57**

Computing Platform

Return to Glossary

A **Computing Platform** is the environment in which a piece of software is executed. It may be the hardware or the Operating System (OS), even a web browser and associated application programming interfaces, or other underlying software, as long as the program code is executed with it. Computing platforms have different abstraction levels, including a computer architecture, an OS, or runtime libraries. A computing platform is the stage on which computer programs can run.

A platform can be seen both as a constraint on the software development process, in that different platforms provide different functionality and restrictions; and as an assistant to the development process, in that they provide low-level functionality ready-made. For example, an OS may be a platform that abstracts the underlying differences in hardware and provides a generic command for saving files or accessing the network.

Source: [https://en.wikipedia.org/wiki/Computing_platform]]

Conceptual Schema

Return to Glossary

The **Conceptual Schema** offers a single, integrated view of the important information content elements, as well as the major interconnections among them. It is not tied to any specific modeling or definition language; rather, it is sufficiently general that it can be used in conjunction with a wide variety of such languages. The schema is based on fundamental, common notions of processes, and employs simple, natural language terminology. [14]

Source: https://www.computer.org/csdl/proceedings-article/spcon/1994/00344419/12OmNrJAe2C

[14] J. W. Armitage, et al., Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA, USA, Volume 1, 1994, A conceptual schema for process definitions and models, Accessed: 21 May 2021, https://www.computer.org/csdl/proceedings-article/spcon/1994/00344419/12OmNrJAe2C

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:conceptual_schema

Last update: **2021/06/15 08:58**

Condition

Return to Glossary

Condition is …

Source: URI

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:condition

Last update: **2021/06/15 08:58**

Confidentiality

Return to Glossary

See 4.2.4.1 Confidentiality

Confidentiality is the degree to which a product or system ensures that data are accessible only to those authorized to have access. Integrity - Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&start=6

Confidentiality Agreement

Return to Glossary

A **Confidentiality Agreement** is a legal agreement that binds one or more parties to non-disclosure of confidentiality or proprietary information. A confidentiality agreement is often used in situations wherein sensitive corporate information or proprietary knowledge is not to be made available to the general public or to competitors. A Non-Disclosure Agreement (NDA) is a particular type of confidentiality agreement.

Source: https://www.investopedia.com/terms/c/confidentiality_agreement.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:confidentialityagreement

Last update: **2020/11/15 19:56**

Configuration Management (CM)

Return to Glossary

Configuration Management (CM) is a process for maintaining computer systems, servers, and software in a desired, consistent state. It’s a way to make sure that a system performs as expected as changes are made over time.

Source: Configuration Management (CM)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cm

Last update: **2020/11/13 13:03**

Conformance Specification

Return to Glossary

Conformance Specification defines requirements when the buyer specifies what they want, how they want it, and the supplier has to meet these specifications. In contraast, see Performance or Functional Specifications.

Examples of conformance specification are:

- Engineering drawing
- A chemical formula
- Model name, number or brand etc

Source: http://zeritenetwork.com/typesofspecificationswhencontracting/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:conformancespecification

Last update: **2021/06/15 08:59**

Consensus Algorithm

Return to Glossary

A **Consensus Algorithm** is a process in computer science used to achieve agreement on a single data value among distributed processes or systems. Consensus algorithms are designed to achieve reliability in a network involving multiple unreliable nodes. Solving that issue – known as the consensus problem – is important in distributed computing and multi-agent systems.

Source: Consensus Algorithm - Tech Target, April 2018

Container

Return to Glossary

a **Container** is a class, a data structure, or an Abstract Data Type (ADT) whose instances are collections of other objects. In other words, they store objects in an organized way that follows specific access rules. The size of the container depends on the number of objects (elements) it contains. Underlying (inherited) implementations of various container types may vary in size and complexity, and provide flexibility in choosing the right implementation for any given scenario.

Source: Container

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:container

Last update: **2020/11/13 13:03**

Container Engine

Return to Glossary

A **Container Engine** is a piece of software that accepts user requests, including command line options, pulls images, and from the end user's perspective runs the container. There are many container engines, including docker, RKT, CRI-O, and LXD. Also, many cloud providers, Platforms as a Service (PaaS), and Container Platforms have their own built-in container engines which consume Docker or OCI compliant Container Images. Having an industry standard Container Image Format allows interoperability between all of these different platforms.

Source: https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:containerengine

Last update: **2021/06/15 08:59**

Container Host

Return to Glossary

A **Container Host* or** Host OS **is the Host Operating System on which the Container client and Container daemon run. In the case of Linux and non-Hyper-V containers, the Host OS shares its kernel with running containers. For Hyper-V each container has its own Hyper-V kernel. Source: http://www.floydhilton.com/docker/2017/03/31/Docker-ContainerHost-vs-ContainerOS-Linux-Windows.html ~~DISCUSSION:on|Outstanding Issues~~ ~~DISCUSSION:off~~**

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:container_host

Last update: **2021/06/15 09:00**

Container OS

Return to Glossary

Container OS or **Base OS** refers to an image that contains an operating system such as Ubuntu, CentOS, or windowsservercore. Typically, the local image is built on top of a Base OS image so that it can take use parts of the OS. Note that windows containers require a Base OS, while Linux containers do not.

Source: http://www.floydhilton.com/docker/2017/03/31/Docker-ContainerHost-vs-ContainerOS-Linux-Windows.html

Control Level

Return to Glossary

The **Control Level** is the second level up in the Automation Pyramid. It uses the the control devices to "run" the devices in the Field Level. The Control Devices make decisions based on information provided by sensors, switches, and other input devices to complete the programmed task.

Source: Control Level

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:control_level

Last update: **2020/11/15 20:15**

Copyleft

Return to Glossary

Copyleft refers to licenses that allow derivative works but require them to use the same license as the original work. For example, if you write some software and release it under the GNU General Public License (a widely-used copyleft license), and then someone else modifies that software and distributes their modified version, the modified version must be licensed under the GNU GPL too — including any new code written specifically to go into the modified version. Both the original and the new work are Open Source; the copyleft license simply ensures that property is perpetuated to all downstream derivatives. (There is at least one copyleft license, the Affero GPL, that even requires you to offer the source code, under the AGPL, to anyone to whom you make the software's functionality available as a network service — however, most copyleft licenses activate their share-and-share-alike requirement on distribution of a copy of the software itself. You should read the license to understand its requirements for source code distribution.)

Most copyleft licenses are Open Source, but not all Open Source licenses are copyleft. When an Open Source license is not copyleft, that means software released under that license can be used as part of programs distributed under other licenses, including proprietary (non-open-source) licenses. For example, the BSD license is a non-copyleft Open Source license. Such licenses are usually called either “non-copyleft” or “permissive” open source licenses

Copyleft provisions apply only to actual derivatives, that is, cases where an existing copylefted work was modified. Merely distributing a copyleft work alongside a non-copyleft work does not cause the latter to fall under the copyleft terms.

Source: What is "copyleft"? Is it the same as "open source"?

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:copyleft

Last update: **2020/11/13 13:02**

Copyright

Return to Glossary

Copyright refers to the legal right of the owner of Intellectual Property (IP). In simpler terms, copyright is the right to copy. This means that the original creators of products and anyone they give authorization to are the only ones with the exclusive right to reproduce the work.

Copyright law gives creators of original material the exclusive right to further use and duplicate that material for a given amount of time, at which point the copyrighted item becomes public domain.

Source: https://www.investopedia.com/terms/c/copyright.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:copyright

Last update: **2021/06/15 09:02**

Cryptocurrency

Return to Glossary

A **Cryptocurrency** is a digital or virtual currency that is secured by cryptography, which makes it nearly impossible to counterfeit or double-spend. Many cryptocurrencies are decentralized networks based on blockchain technology—a distributed ledger enforced by a disparate network of computers.

Source: https://www.investopedia.com/terms/c/cryptocurrency.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:c:cryptocurrency

Last update: **2021/06/15 09:03**

CyberSecurity Culture (CSC)

Return to Glossary

CyberSecurity Culture (CSC) of organizations refers to the knowledge, beliefs, perceptions, attitudes, assumptions, norms and values of people regarding CyberSecurity and how they manifest in people's behaviour with information technologies. CSC is about making information security considerations an integral part of an employee's job, habits and conduct,embedding them in their day-to-day actions.Adopting the right approach to information security enables a resilient CSC to develop naturally from the behaviours and attitudes of employees towards information assets at work,1and as part of a company's wider organisational culture, its CSC can be shaped, directed and transformed.2However, business environments constantly change, hence organisations must actively maintain and adapt their CSC in response to new technologies and threats, as well as their changing goals, processes and structures. A successful CSC shapes the security thinking of all staff (including the security team), improving resilience against all cyber threats, especially when initiated through social engineering,3while avoiding imposing burdensome security steps that prevent staff from effectively performing their key business functions.

Source: https://www.enisa.europa.eu/publications/cyber-security-culture-in-organisations

D

Return to Glossary

Create a Glossary entry starting with 'D' **Word or Expression** →	□	Add page

- Daemon
- Data as a Service (DaaS)
- Data at Rest
- Data Definition Language (DDL)
- Data Distribution Service (DDS)
- Data in Motion
- Data in Use
- Data Integrity
- Data Link Layer (DLL)
- Data Logging
- Data Manipulation Language (DML)
- Data Model (DM)
- Data Object (DO)
- Data Protection
- Data Protection Act 2018
- Data Quality
- Data Reader
- Data Security
- Data Structure
- Data Writer
- Data-Centric
- Data-Centric Publish-Subscribe (DCPS)
- Database Driver
- DataBase Management System (DBMS)
- Datastore
- DDS Domain
- de facto Standard
- Decentralized Finance (DeFi)
- Delegated Byzantine Fault Tolerant (dBFT)
- Delegated Proof of Stake (DPoS)
- Department of Defense (DoD)
- Dependent Event
- DevOps
- DIDO Domain Community
- DIDO Ecosphere Community
- DIDO Ecosystem Community
- DIDO Platform
- Digital Rights

- Digital Rights Management (DRM)
- Directed Acyclic Graph (DAG)
- Disconnected, Intermittent and Limited (DIL)
- Discovery
- Disk Image
- Distributed Application (ĐApp or DApp)
- Distributed Denial-of-Service (DDoS)
- Distributed Immutable Data Objects (DIDO)
- Distributed Ledger Technology (DLT)
- Distributed System
- Docker
- Document Object Model (DOM)
- Domain Integrity
- Domain Knowledge
- Domain Name System (DNS)
- Domain Participant
- Download Speed
- Downtime
- Driver's Privacy Protection Act of 1994 (DPPA)
- Duck Typing
- Durability
- Dynamic Host Configuration Protocol (DHCP)
- Dynamic Link Library (.dll)

=-
=-

Daemon

Return to Glossary

Daemon is a process that runs in the background and performs a specified operation at predefined times or in response to certain events.

The term daemon is a UNIX term, though many other operating systems provide support for daemons, though they're sometimes called other names. Windows, for example, refers to daemons as System Agents and services.

Typical daemon processes include print spoolers, e-mail handlers, and other programs that perform administrative tasks for the operating system. The term comes from Greek mythology, where daemons were guardian spirits.

Source: https://www.webopedia.com/TERM/D/daemon.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:daemon

Last update: **2020/11/13 03:02**

Data-Centric

Return to Glossary

A **Data-Centric** environment allows you to have a communication mechanism that is custom-tailored to your distributed application’s specific requirements. Distributed application developers can concentrate on the operation of their specific application—without worrying about how they are going to communicate with the other applications in the environment.

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:data_centric

Last update: **2021/06/15 11:01**

Data-Centric Publish-Subscribe (DCPS)

Return to Glossary

Data-Centric Publish-Subscribe (DCPS) is the portion of the Data Distribution Service (DDS) Specification that addresses Data-Centric publish-subscribe communications. The Data Distribution Service (DDS) standard defines a language-independent model of publish-subscribe communications that has standardized mappings into various implementation languages. Connext DDS offers C, Traditional C++, Modern C++, C++/CLI, C#, and Java versions of the DCPS API.

The publish-subscribe approach to distributed communications is a generic mechanism that can be employed by many different types of applications. The DCPS model described in this chapter extends the publish-subscribe model to address the specific needs of real-time, data-critical applications. As you'll see, it provides several mechanisms that allow application developers to control how communications works and how the middleware handles resource limitations and error conditions.

The “data-centric” portion of the term DCPS describes the fundamental concept supported by the design of the API. In data-centric communications, the focus is on the distribution of data between communicating applications. A data-centric system is comprised of data publishers and data subscribers. The communications are based on passing data of known types in named streams from publishers to subscribers.

In contrast, in object-centric communications the fundamental concept is the interface between the applications. An interface is comprised of a set of methods of known types (number and types of method arguments). An object-centric system is comprised of interface servers and interface clients, and communications are based on clients invoking methods on named interfaces that are serviced by the corresponding server.

Data and object-centric communications are complementary paradigms in a distributed system. Applications may require both. However, real-time communications often fit a data-centric model more naturally.

Source: Data-Centric Publish-Subscribe (DCPS)

Data as a Service (DaaS)

Return to Glossary

Data as a service (DaaS) is a cloud strategy used to facilitate the accessibility of business-critical data in a well-timed, protected and affordable manner. DaaS depends on the principle that specified, useful data can be supplied to users on demand, irrespective of any organizational or geographical separation between consumers and providers.

DaaS eliminates redundancy and reduces associated expenditures by accommodating vital data in a single location, allowing data use and/or modification by multiple users via a single update point. Initially used in Web mashups, the DaaS strategy is often used by commercial organizations.

Source: Data as a Service (DaaS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:daas

Last update: **2020/11/13 03:02**

Data at Rest

Return to Glossary

Data at Rest refers to data stored on a device or backup medium in any form. It is data at rest is inactive data not currently being transmitted across a network or actively being read or processed and is in a stable state. It is not traveling within the system or network, and it is not being acted upon by any application or the Central Processing Unit (CPU).

Source: Data at Rest

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dataatrest

Last update: **2020/11/15 15:07**

Database Driver

Return to Glossary

A **Database Driver** is a computer program that implements a protocol (Open Database Connectivity (ODBC) or Java Database Connectivity(JDBC)) for a database connection. The driver works like an adaptor which connects a generic interface to a specific database vendor implementation. … To connect with individual databases, JDBC requires drivers for each specific database type

Source: https://www.jdatalab.com/information_system/2017/02/16/database-driver.html

DataBase Management System (DBMS)

Return to Glossary

A DataBase Management System (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data.

A DBMS relieves users of framing programs for data maintenance. Fourth-generation query languages, such as SQL, are used along with the DBMS package to interact with a database.

Source: https://www.techopedia.com/definition/24361/database-management-systems-dbms

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dbms

Last update: **2020/11/13 03:02**

Data Definition Language (DDL)

Return to Glossary

A **Data Definition Language (DDL)** is a computer language used to create and modify the structure of database objects in a database. These database objects include views, schemas, tables, indexes, etc.

This term is also known as data description language in some contexts, as it describes the fields and records in a database table.

Source:
https://www.techopedia.com/definition/1175/data-definition-language-ddl#techopedia-explains-data-definition-language-ddl

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:ddl

Last update: **2021/06/15 11:02**

Data Distribution Service (DDS)

Return to Glossary

DDS is a specification for publish-subscribe data-distribution systems. The purpose of the specification is to provide a common application-level interface that clearly defines the data distribution service. The specification describes the service using Unified Modeling Language (UML), thus providing a platform-independent model that can then be mapped into a variety of concrete platforms and programming languages.

The Object Management Group® (OMG) DDS attempts to unify the common practice of several existing implementations enumerating and providing formal definitions for the Quality of Service (QoS) settings that can be used to configure the service.

Source: https://community.rti.com/glossary/dds

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dds

Last update: **2021/06/15 11:03**

Data in Motion

Return to Glossary

Data in Motion is data that is currently traveling across a network or sitting in a computer's Reliability, Maintainability, and Availability (RAM) ready to be read, updated, or processed. Data crossing over networks from local to cloud storage or from a central mainframe to a remote terminal should be encrypted so that it cannot be read or manipulated by any machine or hacker between the data's source and destination. This data in motion includes data moving across a cables and wireless transmission. It can be emails or files transferred over File Transfer Protocol (FTP) or Secure Shell (SSH).

Source: https://aspg.com/2014/09/03/three-states-digital-data/#.Xv_LBfJ7lgQ

Data Integrity

Return to Glossary

See 4.2.4.2 Data Integrity

Data Integrity is the overall completeness, accuracy and consistency of data. This can be indicated by the absence of alteration between two instances or between two updates of a data record, meaning data is intact and unchanged. Data integrity is usually imposed during the database design phase through the use of standard procedures and rules. Data integrity can be maintained through the use of various error-checking methods and validation procedures.

Source: Data Integrity

Data in Use

Return to Glossary

Data in Use is data that is not just being stored passively on a hard drive or external storage media. This is data that is being processed by one or more applications. This is data currently in the process of being generated, updated, appended, or erased. It also includes data being viewed by users accessing it through various endpoints. Data in use is susceptible to different kinds of threats depending on where it is in the system and who is able to use it. The most vulnerable point for data in use is at the endpoints where users are able to access and interact with it.

Source: https://aspg.com/2014/09/03/three-states-digital-data/#.Xv_LBfJ7lgQ

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:data_in_use

Last update: **2021/06/15 11:04**

Data Link Layer (DLL)

Return to Glossary

The **Data Link Layer (DLL)** is fifth layer in the Open Systems Interconnection (OSI) Model and is used for the encoding, decoding and logical organization of data bits. Data packets are framed and addressed by this layer, which has two sublayers.

The Data Link Layer's first sublayer is the media access control (MAC) layer. It is used for source and destination addresses. The MAC layer allows the data link layer to provide the best data transmission vehicle and manage data flow control.

The Data Link Layer's second sublayer is the logical link control. It manages error checking and data flow over a network.

Source: Data Link Layer

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:datalinklayer

Last update: **2020/11/14 16:20**

Data Logging

Return to Glossary

Data Logging is the process of collecting and storing data over a period of time in order to analyze specific trends or record the data-based events/actions of a system, network or IT environment. It enables the tracking of all interactions through which data, files or applications are stored, accessed or modified on a storage device or application.

Source: Data Logging

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:datalog

Last update: **2020/11/14 17:56**

Data Manipulation Language (DML)

Return to Glossary

A **Data Manipulation Language (DML)** is a family of computer languages including commands permitting users to manipulate data in a database. This manipulation involves inserting data into database tables, retrieving existing data, deleting data from existing tables and modifying existing data. DML is mostly incorporated in SQL databases.

Source: https://www.techopedia.com/definition/1179/data-manipulation-language-dml

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dml

Last update: **2021/06/15 11:05**

Data Model (DM)

Return to Glossary

A **Data Model (DM)** refers to the logical inter-relationships and data flow between different data elements involved in the information world. It also documents the way data is stored and retrieved. Data models facilitate communication business and technical development by accurately representing the requirements of the information system and by designing the responses needed for those requirements. Data models help represent what data is required and what format is to be used for different business processes.

Source: Data Model (DM)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dm

Last update: **2020/11/13 03:03**

Data Object (DO)

Return to Glossary

Data Object (DO) is an Object that is an instance of a Class of things used within a system. In Distributed Computing, the Data Object can either represent the class of things used by the Distributed Systems to support the distributed computing (i.e., transactions, blocks, etc.) or they can represent the class of things that will be distributed across the distributed computing network (i.e., supplies, cryptocurrencies, vehicle registrations, etc.).

Source: Reference Architecture (RA)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:data_object

Last update: **2021/06/15 11:05**

Data Protection

Return to Glossary

Data Protection is the process of safeguarding important information from corruption, compromise or loss.

Source: Data Protection

Data Protection Act 2018

Return to Glossary

Data Protection Act 2018 controls how your personal information is used by organisations, businesses or the government. It is the United Kingdom's implementation of the General Data Protection Regulation (GDPR).

Under the Data Protection Act 2018, you have the right to find out what information the government and other organisations store about you. These include the right to:

bEveryone responsible for using personal data has to follow strict rules called ‘data protection principles’. They must make sure the information is:

- used fairly, lawfully and transparently
- used for specified, explicit purposes
- used in a way that is adequate, relevant and limited to only what is necessary
- accurate and, where necessary, kept up to date
- kept for no longer than is necessary
- handled in a way that ensures appropriate security, including protection against unlawful or unauthorised processing, access, loss, destruction or damage

There is stronger legal protection for more sensitive information, such as:

- race
- ethnic background
- political opinions
- religious beliefs
- trade union membership
- genetics
- biometrics (where used for identification)
- health
- sex life or orientation

There are separate safeguards for personal data relating to criminal convictions and offences.

Source: https://www.gov.uk/data-protection

Data Quality

Return to Glossary

Data Quality refers to the state of qualitative or quantitative pieces of information. There are many definitions of data quality, but data is generally considered high quality if it is “fit for [its] intended uses in operations, decision making and planning”. Moreover, data is deemed of high quality if it correctly represents the real-world construct to which it refers.

Source: Data Quality

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dataquality

Last update: **2020/11/15 21:25**

Data Reader

Return to Glossary

An entity attached to a Subscriber, used to subscribe to a Topic, providing type-safe operations to read/receive data. Each Data Reader can be associated to only one Topic and therefore exactly one data type. This Topic must be created before the Data Reader. A data reader can obtain its subscribed data via two approaches:

- Listener-based approach, provides an asynchronous mechanism to obtain data via call-backs in a separate thread that does not block the main application.
- WaitSet-based Approach – A synchronous approach, blocks the current thread until one or more conditions attached to the WaitSet are met, at which point a list of active_conditions are returned.

There are several ways samples can be read using a data reader, this includes reading with conditions and some buffer management, described in greater detail in your respective language reference guide.

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:data_reader

Last update: **2021/06/15 11:06**

Data Security

Return to Glossary

Data Security refers to the process of protecting data from unauthorized access and data corruption throughout its lifecycle. Data security includes data encryption, hashing, tokenization, and key management practices that protect data across all applications and platforms.

Source: Data Security

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:datasecurity

Last update: **2020/11/15 14:58**

Datastore

Return to Glossary

A **Datastore** is a repository for storing, managing and distributing data sets on an enterprise level. It is a broad term, which incorporates all types of data that is produced, stored and used by an organization. The term refers specifically to data that is at rest and used by one or more data-driven applications, services or individuals. A database is a kind of **datastore**, but there are datastores that are not databases, for example W3C documents or JSON files.

Source: Datastore

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:datastore

Last update: **2020/11/13 03:03**

Data Structure

Return to Glossary

Data Structure refers to methods of organizing units of data within larger data sets. Achieving and maintaining specific data structures help improve data access and value. Data structures also help programmers implement various programming tasks.

Source: Data Structure

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:datastructure

Last update: **2020/11/13 13:04**

Data Writer

Return to Glossary

An entity attached to a Publisher bound to publish samples from only one Topic and therefore exactly one data type, providing type-safe operations to write/send data. The Topic must be created before the Data Writer.

Source: OpenSplice Glossary

DDS Domain

Return to Glossary

A **DDS Domain** is a communication context, which provides a virtual environment, encapsulating different concerns and thereby optimizing communications. Data Distribution Service (DDS) applications send and receive data within a domain, which provides a virtual communication environment for participants having the same domain Identifier (ID). This environment also isolates participants associated with different domains, i.e., only participants within the same domain can communicate, which is useful for isolating and optimizing communication within a community that shares common interests.

Source: DDS Domain

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:domain_dds

Last update: **2021/06/15 11:07**

Decentralized Finance (DeFi)

Return to Glossary

Decentralized Fiinance (DeFi) is a blockchain-based form of finance that does not rely on central financial intermediaries such as brokerages, exchanges, or banks to offer traditional financial instruments, and instead utilizes smart contracts on blockchains, the most common being Ethereum. DeFi platforms allow people to lend or borrow funds from others, speculate on price movements on a range of assets using derivatives, trade cryptocurrencies, insure against risks, and earn interest in savings-like accounts. DeFi uses a layered architecture and highly composable building blocks. Some DeFi applications promote high interest rates but are subject to high risk. By October 2020, over $11 billion (worth in cryptocurrency) was deposited in various decentralized finance protocols, which represented more than a tenfold growth during the course of 2020. As of January 2021, approximately $20.5 billion was invested in DeFi.

Source: https://en.wikipedia.org/wiki/Decentralized_finance

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:defi

Last update: **2021/06/15 11:08**

de facto Standard

Return to Glossary

A de facto standard is something that is used so widely that it is considered a standard for a given application although it has no official status. Some examples are libraries, components, or systems that are offered by a proprietary corporation that have not gone through a Standards Developing Organization (SDO). Examples of *de facto* standards are Ethereum, Windows, Java, Log4J, etc.

Source: De facto Standard

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:defactostd

Last update: **2020/11/13 03:02**

Delegated Byzantine Fault Tolerant (dBFT)

Return to Glossary

Delegated Byzantine Fault Tolerant (dBFT) is a consensus mechanism that was made popular by a cryptocurrency called NEO. dBFT essentially works in a similar fashion to a country's governance system, having its own citizens, delegates, and speakers to ensure that the country (Node Network) is functional. The method is closer to Proof of Stake (PoS) rather than Proof of Work (PoW), by utilizing a voting system to choose delegates and speaker.

Source: Delegated Byzantine Fault Tolerant (dBFT)

Delegated Proof of Stake (DPoS)

Return to Glossary

Delegated Proof of Stake (DPoS) is an alternative consensus algorithm to the Proof of Work (PoW) or Proof of Stake (PoS) algorithms. DPoS is a system in which a fixed number of elected entities (called Block Producers or witnesses) are Block Validators selected to create blocks in a round-robin order. Block Producers are voted into power by the users of the Node Network, that each get a number of votes proportional to the number of tokens they own on the network (their stake).

Alternatively, voters can choose to delegate their stake to another voter, who will vote in the Block Producer election on their behalf.

> **Note:** DPoS is a protocol that sacrifices decentralization for throughput due to the low number of Block Producers.

Source: Delegated Proof of Stake DPoS

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:delegated_proof_of_stake_dpos

Last update: **2020/11/13 03:03**

Department of Defense (DoD)

Return to Glossary

The United States **Department of Defense (DoD)** is the department within the United States that provides the military forces needed to deter war, and to protect the security of the United States. Source: Department of Defense (DoD)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dod

Last update: **2020/11/13 13:04**

Dependent Event

Return to Glossary

A **Dependent Event** is when one event influences the outcome of another event in a probability scenario.

Source: Dependent Event

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dependevent

Last update: **2020/11/14 13:40**

DevOps

Return to Glossary

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from Agile methodology.

Source: DevOps

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:devops

Last update: **2021/06/15 11:10**

DIDO Domain Community

Return to Glossary

DIDO Domain Community is the lowest level Community of Interest (CoI). The Domain has a sub-charter approved by the DIDO Ecosystem Community. The Domain usually relies on the Ecosphere for Bylaws and Policies and Procedures (P&P) but can provide addendums that do not conflict with the Ecosphere. The primary role of the Domain is to produce a product which meets the Functional and Non-Functional Requirements of the Ecosystem and the Ecosphere. As a general rule, the Domain actually builds or deploys things to be integrated into the Ecosystem. The Domain may have more Intellectual Property (IP) Rights than the Ecosystem. It can have a subset the Copyrights allowed by the Ecosystem.

The Domain role is to build products as per the requirements and maintain products according to the bug tracking system. The Domain is responsible for all testing at the Domain level (See: 4.3.3.5 Testability).

See also:

- DIDO Ecosphere Community
- DIDO Ecosystem Community

Source: https://www.omgwiki.org/dido/doku.php?id=dido:public:ra

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dido_domain_community

Last update: **2021/06/15 11:13**

DIDO Ecosphere Community

Return to Glossary

DIDO Ecosphere Community is the highest level Community of Interest (CoI) that encapsulates DIDO Ecosystem Communities and DIDO Domain Communities. The Ecosphere usually provides high level requirements and some funding for the administration of the other CoIs. The Ecosphere role is to act as a coordinator of the Ecosystems and to provide a framework for all other CoIs to establish working agreements such as Memorandum of Agreement (MOA) or Memorandum of Understanding (MOUs). The Ecosphere is often the only CoI that is recognized as a Legal Entity with legally binding Charter, Bylaws and official Policies and Procedures. Often the Ecosphere control Intellectual Property Rights (IP) and allowable Copyrights that are acceptable for the Ecosphere and the Domain.

See also:

- DIDO Ecosystem Community
- DIDO Domain Community

Source: Reference Architecture (RA)

DIDO Ecosystem Community

Return to Glossary

DIDO Ecosystem Community is the midlevel level Community of Interest (CoI) that encapsulates DIDO Domain Communities. The Ecosystem has a sub- charter] approved by the DIDO Ecosphere Community. The Ecosystem usually relies on the Ecosphere for Bylaws and Policies and Procedures (P&P) but can provide addendums that do not conflict with the Ecosphere. The primary role of the Ecosystem is to coordinate the activities of the Domains which fall under its jurisdiction. As a general rule, the Ecosystem does not actually create anything but acts as the integrator and coordinator all the Domains it is responsible for. The Ecosystem may have more restrictive Intellectual Property (IP) Rights than the Ecosphere. It can only subset the Copyrights allowed by the Ecosphere.

The Ecosphere role is to act as a coordinator of the Domains, however, one Ecosystem can also have a Sub-Ecosysem that it is responsible for. The Ecosystem can have its own bug tracking system that covers integration issues. The Ecosystem is responsible for all integration testing.

See also:

- DIDO Ecosphere Community
- DIDO Domain Community

Source: https://www.omgwiki.org/dido/doku.php?id=dido:public:ra

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dido_ecosystem_community

Last update: **2021/06/15 12:38**

DIDO Platform

Return to Glossary

DIDO Platform is a combination of operating system, runtime, language platforms and the specific DIDO software (i.e., Ethereum, Hyperledger, Itota, etc.).

Source: Reference Architecture (RA)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dido_platform

Last update: **2021/06/15 12:39**

Digital Rights

Return to Glossary

Digital Rights are the rights of individuals as it pertains to computer access and the ability to use, create and publish digital media. Digital Rights can also refer to allowed permissions for fair use of digital copyrighted materials.

Digital Rights are extensions of human rights like freedom of expression and the right to privacy. The extent to which Digital Rights are recognized varies from country to country, but Internet access is a recognized right in several countries.

Digital Rights Management (DRM) software is often used to manage or control what is done with copyrighted files. DRM tends to be unpopular with users as it can complicate and limit access to digital media. This restriction can make piracy more tempting.

Source: https://whatis.techtarget.com/definition/digital-rights

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:digital_rights

Last update: **2021/06/15 12:39**

Digital Rights Management (DRM)

Return to Glossary

Digital Rights Management (DRM) is the protection of copyrighted works by various means to control or prevent digital copies from being shared over computer networks or telecommunications networks.

The digitalization of content has challenged traditional copyright laws on two fronts. First, it has enabled nearly cost-free reproduction and large-scale distribution of digital content. Second, existing digital content easily can be remixed and “mashed-up” (combined in various ways) with other content to produce new works. In response to these changes, copyright holders have sought greater protection through legal and technological remedies.

Source: https://www.britannica.com/topic/digital-rights-management

Directed Acyclic Graph (DAG)

Return to Glossary

Directed Acyclic Graph (DAG) mathematically a DAG is a graph that travels in one direction without cycles connecting the other edges. This means it is impossible to traverse the entire graph starting at one edge. The edges of the directed graph only go one way. The graph is a topological sorting, where each node is in a certain order.

Imagine a collection of individual transactions where each transaction is linked to at least one other transaction in the following way:

Directed

The links point in the same direction with earlier transactions linked to later transactions, and so on.

Acyclic

Loops are not possible. A transaction cannot loop back on itself after linking to another transaction.

Graph

The mesh of connected transactions can be represented as nodes in a graph network, in which nodes are joined to each other by links.[15]

[15] “What is DAG Distributed Ledger Technology?”, Max Thake, 9 November 2018, https://medium.com/nakamo-to/what-is-dag-distributed-ledger-technology-8b182a858e19

Disconnected, Intermittent and Limited (DIL)

Return to Glossary

Disconnected, Intermittent and Limited (DIL) occur in wireless/mobile networking environments, e.g. rural area networks, vehicular networks, battlefield networks and other resource-constrained or disadvantaged networks. Efficient and effective interoperability in these networks is highly demanded for various mission-critical application scenarios such as disaster relief in ravaged regions, search and rescue in remote areas and military/tactical operations in hostile environments.

Source: Disconnected, Intermittent and Limited (DIL)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dil

Last update: **2020/11/13 13:04**

Discovery

Return to Glossary

Discovery is typically define in the context of networks. Thus the term “discovery” refers the process through which computers and network devices are able to find each other. It enables network devices over a local network to connect and communicate with other devices. In the context of OMG's DDS, “Discovery” refers to the process by which a DomainParticipant learns about other DomainParticipants, DataWriters, and DataReaders within the same domain by detecting: the presence of new entities in the Domain, that the entities have shut down cleanly, and entities have timed out because of lack of communication.

Sources: https://www.techopedia.com/definition/29965/network-discovery, https://community.rti.com/glossary/discovery

c

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:discovery

Last update: **2021/06/15 13:07**

Disk Image

Return to Glossary

Disk Image is a single file or storage device that holds a replica of all data on a storage medium or device, such as a hard drive, tape drive, CD, DVD, floppy disk or key drive. A disk image is usually created through a sector-by-sector replication of the original - or source - storage medium, including the structure (directories and folders) and contents (files).

Source: https://www.techopedia.com/definition/12705/disk-image

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:disk_image

Last update: **2021/06/15 13:08**

Distributed Application (ĐApp or DApp)

Return to Glossary

Distributed Application (DApp or DApp) is software that is executed or run on multiple computers within a network. These applications interact in order to achieve a specific goal or task. Traditional applications relied on a single system to run them. Even in the Client-Server model, the application software had to run on either the client, or the server that the client was accessing. However, distributed applications run on both simultaneously. With distributed applications, if a node that is running a particular application goes down, another node can resume the task.

Source: Distributed Application (Dapp)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dapp

Last update: **2020/11/13 03:03**

Distributed Denial-of-Service (DDoS)

Return to Glossary

A **Distributed Denial-of-Service (DDoS)** is malicious attack attempting to disrupt normal traffic of a targeted Server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic.

DDoS attacks achieve effectiveness by utilizing multiple compromised computer systems as sources of attack traffic. Exploited machines can include computers and other networked resources such as Internet of Things (IOT) devices.

Source: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:ddos

Last update: **2021/06/15 13:08**

Distributed Immutable Data Objects (DIDO)

Return to Glossary

The term Distributed Immutable Data Objects (DIDO) refers to the underlying technologies supporting distributed data and computation across a distributed network of peers using consensus algorithms to maintain integrity and consistency across the network.

Source: Distributed Immutable Data Objects (DIDO)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dido

Last update: **2020/11/13 13:05**

Distributed Ledger Technology (DLT)

Return to Glossary

Distributed Ledger Technology (DLT) is a digital system for recording the transaction of assets in which the transactions and their details are recorded in multiple places at the same time. Unlike traditional databases, distributed ledgers have no central data store or administration functionality.

Source: Distributed Ledger Technology (DLT)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dlt

Last update: **2020/11/13 03:02**

Distributed System

Return to Glossary

A **Distributed System** (**Distributed Computing** and **Distributed Database**), in its most simplest definition is a group of computers communicating through messages working together to behave as a single computer to the end-user.

These machines have a shared state, operate concurrently and can fail independently without affecting the whole system's uptime.

Source: Distributed System

Docker

Return to Glossary

Docker is an open platform that helps with the universal distribution of applications. It has become a standard for certain types of container virtualization systems and has been adopted by various companies as a software container strategy.

See also https://docs.docker.com/

Source: https://www.techopedia.com/definition/31115/docker

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:docker

Last update: **2020/11/14 17:49**

Domain Integrity

Return to Glossary

Domain Integrity is the collection of processes that ensure the accuracy of each piece of data in a domain. In this context, a domain is a set of acceptable values that a column is allowed to contain. It can include constraints and other measures that limit the format, type, and amount of data entered.

Source: Domain Integrity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:domaininegrity

Last update: **2020/11/15 21:19**

Domain Knowledge

Return to Glossary

Domain Knowledge is knowledge of a specific, specialized discipline or field, in contrast to general knowledge, or domain-independent knowledge. The term is often used in reference to a more general discipline, as, for example, in describing a software engineer who has general knowledge of programming, as well as domain knowledge about the pharmaceutical industry. People who have domain knowledge, are often considered specialists or experts in the field.

Source: Domain Knowledge

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:domain_knowledge

Last update: **2020/11/14 20:29**

Domain Name System (DNS)

Return to Glossary

Domain Name System (DNS) is a hierarchical naming system built on a distributed database. This system transforms domain names to IP addresses and makes it possible to assign domain names to groups of Internet resources and users, regardless of the entities' physical location.

Source: Domain Name System (DNS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:dns

Last update: **2020/11/13 03:03**

Domain Participant

Return to Glossary

A **Domain Participant** is an entity that represents a Data Distribution Service (DDS) application’s participation in a domain. It serves as factory, container, and manager for the DDS entities.

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:domain_participant

Last update: **2020/11/16 18:57**

Download Speed

Return to Glossary

Download Speed refers to how many megabits of data per second it takes to download data from a server in the form of images, videos, text and more. Activities such as listening to music on Spotify, downloading large files or streaming videos on Netflix all require you to download data.

Source: Download Speed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:downloadspeed

Last update: **2020/11/16 21:01**

Downtime

Return to Glossary

Downtime (T_d) is the Mission Duration times the sum of all of the different kinds of time required to transition from being down to the time to be fully operational divided by the Mean Time Between Failure.

- Mean Time To Repair (MTTR) is time required to restore operations the level defined in the system specification
- Mean Logistics Delay Time (MLDT) is time required to obtain parts from the part depot or from the manufacturer including transportation to the site
- Mean Active Maintenance Down Time (MAMDT) is average time required to perform diagnostics and replace parts

Source: Downtime

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:downtime

Last update: **2020/11/14 13:33**

Driver's Privacy Protection Act of 1994 (DPPA)

Return to Glossary

The **Driver's Privacy Protection Act of 1994 (DPPA) (18 U.S. Code § 2721 et seq.)** governs the privacy and disclosure of personal information gathered by state Departments of Motor Vehicles, including photographs, Social Security Number (SSN), Driver Identification Number (DID), name, address (but not the five-digit ZIP code), telephone number, medical information and disability information.

Source: https://iclg.com/practice-areas/data-protection-laws-and-regulations/usa

Duck Typing

Return to Glossary

Duck Typing is a style of dynamic typing in which an object's current set of methods and properties determines the valid semantics, rather than its inheritance from a particular class or implementation of a specific interface.

Source: https://www.yourdictionary.com/duck-typing

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:ducktyping

Last update: **2020/11/14 18:26**

Durability

Return to Glossary

Durability is …

Source: URI

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:d:durability

Last update: **2021/06/15 13:09**

Dynamic Host Configuration Protocol (DHCP)

Return to Glossary

Dynamic Host Configuration Protocol (DHCP) is a network management Protocol used to dynamically assign an Internet Protocol (IP) address to any device, or node, on a network so it can communicate using IP. DHCP automates and centrally manages these configurations rather than requiring network administrators to manually assign an Internet Protocol address (IP address) to all network devices. DHCP can be implemented on small local networks Local Area Network (LAN), as well as Wide Area Network (WAN) large enterprise networks.

Source: https://searchnetworking.techtarget.com/definition/DHCP

Dynamic Link Library (.dll)

Return to Glossary

A **Dynamic Link Library (.dll)** is a Relocatable Objects]] (i.e., shared program module) with ordered code, methods, functions, enums and structures that may be dynamically called by an executing program during run time. A DLL usually has a file extension ending in .dll. Other file extensions are .drv and .ocx.

DLLs were developed by Microsoft and work only with the Windows operating system (OS).

Source: https://www.techopedia.com/definition/24835/dynamic-link-library-dll

E

Return to Glossary

Create a Glossary entry starting with 'E' **Word or Expression** →		Add page

- Economies of Scale
- Elastic Compute Cloud (EC2)
- Encryption
- End-of-life (EoL)
- End-to-End Solution (E2ES)
- End-to-End Testing (E2E Testing)
- Endianness
- Endpoint
- Enterprise Resource Planning (ERP)
- Entity
- Entity Integrity
- Environment Variables
- Ethereum Improvement Proposal (EIP)
- Ethereum Node
- Ethereum Request for Comment (ERC)
- Ethereum Virtual Machin (EVM)
- Ethernet
- Evidence
- Executable File

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:start

Last update: **2021/06/14 21:34**

Economies of Scale

Return to Glossary

Economies of Scale are cost advantages reaped by companies when production becomes efficient. Companies can achieve economies of scale by increasing production and lowering costs. This happens because costs are spread over a larger number of goods. Costs can be both fixed and variable.

The size of the business generally matters when it comes to economies of scale. The larger the business, the more the cost savings.

Economies of scale can be both internal and external. Internal economies of scale are based on management decisions, while external ones have to do with outside factors.

Source: https://www.investopedia.com/terms/e/economiesofscale.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:economyofscale

Last update: **2020/11/16 21:15**

Elastic Compute Cloud (EC2)

Return to Glossary

Elastic Compute Cloud (EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Source: Elastic Compute Cloud (EC2)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:ec2

Last update: **2020/11/13 15:29**

Encryption

Return to Glossary

Encryption is the process of using an algorithm to transform information to make it unreadable for unauthorized users. This cryptographic method protects sensitive data such as credit card numbers by encoding and transforming information into unreadable cipher text. This encoded data may only be decrypted or made readable with a key. Symmetric-key and asymmetric-key are the two primary types of encryption.

Encryption is essential for ensured and trusted delivery of sensitive information.

Source: https://www.techopedia.com/definition/5507/encryption

End-of-life (EoL)

Return to Glossary

End-of-life (EoL) is a term used by software vendors indicating that it is ending or limiting it’s support on the product and/or version to shift focus on their newer products and/or version. Most vendors define a product lifecycle which consists of several phases during which a product moves where ‘end-of-life’ usually means the last phase (depending on the software vendor). These phases will determine the type of support it’s customers will receive. Examples are maintenance (software updates and security patches) and troubleshooting.

In the specific case of product sales, a vendor may use the more specific term “end-of-sale” (EOS). The timeframe after the last release date depends on the type of product and relates to the expected product lifetime from a customer's point of view.

Source: End-of-life (EoL)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:eol

Last update: **2020/11/13 03:00**

End-to-End Solution (E2ES)

Return to Glossary

An **End-to-End Solution (E2ES)** is a term that means that the provider of an application program, software and system will supply all the software as well as hardware requirements of the customer such that no other vendor is involved to meet the needs. E2ES includes installation, integration, and setup.

Source: https://www.techopedia.com/definition/19057/end-to-end-solution-e2es

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:e2esolution

Last update: **2020/11/15 02:29**

End-to-End Testing (E2E Testing)

Return to Glossary

End-to-End Testing (E2E testing) refers to a software testing method that involves testing an application’s workflow from beginning to end. This method basically aims to replicate real user scenarios so that the system can be validated for integration and Data Integrity.

Essentially, the test goes through every operation the application can perform to test how the application communicates with hardware, network connectivity, external dependencies, databases, and other applications. Usually, E2E testing is executed after functional and system testing is complete.

Source: https://www.browserstack.com/guide/end-to-end-testing

Endianness

Return to Glossary

endianness is the byte order chosen for all digital computing made in a specific computer system and dictates the architecture and low-level programming approach to be used for that system.

Source: Endian

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:endianness

Last update: **2020/11/13 15:29**

Endpoint

Return to Glossary

An **Endpoint** is any device that is physically an end point on a network. Laptops, desktops, mobile phones, tablets, servers, and virtual environments can all be considered endpoints. When one considers a traditional home antivirus, the desktop, laptop, or smartphone that antivirus is installed on is the endpoint.

Source: Endpoint

Enterprise Resource Planning (ERP)

Return to Glossary

Enterprise resource planning (ERP) is a method of efficiently utilizing people, hardware and software to increase productivity and profit, thus simplifying a company’s business processes. ERP may include many software applications or a single (but more complex) software package that smoothly disseminates data required by two or more unique business departments.

Source: Enterprise resource planning (ERP)

Entity

Return to Glossary

The system components are called **Entities** because they all inherit from the Entity class. Each Entity has specialised QoS policies. An Entity may have a Listener, a call back interface for notifications about changes in the Entity’s state or, a wait interface (using WaitSets) for detecting changes in the Entity’s state.

Source: OpenSplice Glossary

Entity Integrity

Return to Glossary

Entity Integrity relies on the creation of primary keys, or unique values that identify pieces of data, to ensure that data isn't listed more than once and that no field in a table is null. It's a feature of relational systems which store data in tables that can be linked and used in a variety of ways.

Source: Entity Integrity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:entityintegrity

Last update: **2020/11/15 21:12**

Environment Variables

Return to Glossary

Environment Variables are values that impact the processes and behavior of running computer systems and OS environments. Running programs may access environment variable values for configuration purposes.

Source: https://www.techopedia.com/definition/15664/environment-variable

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:envvar

Last update: **2020/11/13 15:29**

Ethereum Improvement Proposal (EIP)

Return to Glossary

Ethereum Improvement Proposal (EIP) is a design document providing information to the Ethereum community, or describing a new feature for Ethereum or its processes or environment. The EIP should provide a concise technical specification of the feature and a rationale for the feature. The EIP author is responsible for building consensus within the community and documenting dissenting opinions.

Source: Ethereum Improvement Proposal (EIP)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:eip

Last update: **2020/11/13 15:29**

Ethereum Request for Comment (ERC)

Return to Glossary

Ethereum Request for Comment (ERC) is a technical document used by smart contract developers at Ethereum. They define a set of rules required to implement tokens for the Ethereum ecosystem. These documents are usually created by developers, and they include information about protocol specifications and contract descriptions. Before becoming an standard, an ERC must be revised, commented and accepted by the community through an EIP (Ethereum Improvement Proposal).

Source: Ethereum Request for Comment (ERC)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:erc

Last update: **2020/11/13 15:29**

Ethernet

Return to Glossary

Ethernet is a family of computer networking technologies commonly used in Local Area Network (LAN), metropolitan area networks (MAN) and Wide Area Network (WAN).[1] It was commercially introduced in 1980 and first standardized in 1983 as IEEE 802.3. Ethernet has since retained a good deal of backward compatibility and has been refined to support higher bit rates, a greater number of nodes, and longer link distances.

Source: Ethernet

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:ethernet

Last update: **2020/11/13 15:29**

Evidence

Return to Glossary

Evidence is information used to support a claim. Ideally, evidence should be objective, reproducible, repeatable, and non-disputable. Evidence is key to making a credible assurance case. Without evidence, there is no way to substantiate the claim.

The sources of evidence will depend in part on the availability of artifacts. The evidence data collection may be conducted formally, informally or semiformally.

Evidence comes in many different forms, so it is impossible to dictate what kind of evidence or argument is appropriate for every situation. Evidence may be in the form of an artifact which could be automatically, semi-automatically or manually produced and demonstrated. Evidence must be traceable to its source and method of origination. The evidence may consist of test results, formal analyses, simulation results, hazard analyses, modeling, inspections, and can include deterministic, probabilistic, and qualitative data or information [14]. Examples of evidence data might be software artifacts, methodologies, development processes, testing results, people or programmer expertise and experience credentials, development environments, operational environments, or regulatory compliance.

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4548534/#b13-v115.n03.a05

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:e:evidence

Last update: **2020/11/13 02:59**

Executable File

Return to Glossary

Executable File is a file in a format that the computer can directly execute. Unlike source files, executable files cannot be read by humans. To transform a source file into an executable file, you need to pass it through a Compiler or assembler.

Source: Executable File

F

Return to Glossary

Create a Glossary entry starting with 'F' **Word or Expression →**		Add page

- Failover
- Fault Tolerance
- Federal Deposit Insurance Corporation (FDIC)
- Field Level
- Fifty-One Percent (51% Attack)
- FIGI Symbology
- Figure of Merit (FoM)
- File Transfer Protocol (FTP)
- Financial Instrument Global Identifier (FIGI)
- Firewall
- Five Nines
- Flowchart
- Full Node
- Full-Disk Encryption (FDE)
- Functional Language
- Functional Programming
- Functional Requirements
- Fungible

=-
=-

Failover

Return to Glossary

Failover is the constant capability to automatically and seamlessly switch to a highly reliable backup. This can be operated in a redundant manner or in a standby operational mode upon the failure of a primary server, application, system or other primary system component.

The main purpose of **Failover* is to eliminate, or at least reduce, the impact on users when a system failure occurs.

Source: Failover

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:failover

Last update: **2020/11/14 14:03**

Fault Tolerance

Return to Glossary

See 2.2.2.3 Fault Tolerance

Fault Tolerance is the ability of a system to respond gracefully to an unexpected hardware or software failure. There are many levels of fault tolerance, the lowest being the ability to continue operation in the event of a power failure. Many fault-tolerant computer systems mirror all operations – that is, every operation is performed on two or more duplicate systems, so if one fails the other can take over.

Source: https://www.webopedia.com/TERM/F/fault_tolerance.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:faulttolerance

Last update: **2020/11/14 14:01**

Federal Deposit Insurance Corporation (FDIC)

Return to Glossary

The **Federal Deposit Insurance Corporation (FDIC)** is an independent agency created by the Congress to maintain stability and public confidence in the nation’s financial system. The FDIC insures deposits; examines and supervises financial institutions for safety, soundness, and consumer protection; makes large and complex financial institutions resolvable; and manages receiverships.

Source: https://www.fdic.gov/

Field Level

Return to Glossary

Field Level is lowest level is the Automation Pyramid. It is where products are produced. In other words, this is where the physical work plus monitoring occur. Electric motors, hydraulic and pneumatic actuators to move machinery, proximity switches used to detect that movement or certain materials, photoelectric switches that detect similar things will all play a part in the field level.

Source: Field Level

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:fieldlevel

Last update: **2020/11/15 20:11**

Fifty-One Percent (51% Attack)

Return to Glossary

Fifty-One Percent (51% Attack) refers to an attack on a blockchain – usually bitcoin's, for which such an attack is still hypothetical – by a group of miners controlling more than 50% of the network's mining hashrate, or computing power. The attackers would be able to prevent new transactions from gaining confirmations, allowing them to halt payments between some or all users. They would also be able to reverse transactions that were completed while they were in control of the network, meaning they could double-spend coins.

They would almost certainly not be able to create a create new coins or alter old blocks, so a 51% attack would probably not destroy bitcoin or another blockchain-based currency outright, even if it proved highly damaging.

Source: Fifty-One Percent (51% Attack)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:fifty_one_percent_51_attack

Last update: **2020/11/13 15:29**

FIGI Symbology

Return to Glossary

FIGI symbology consists of the unique, persistent, unchanging alpha-numeric identifier, as well as the multiple individual pieces of associated descriptive metadata.

Source: Standard Symbology for Global Financial Securities

Figure of Merit (FoM)

Return to Glossary

Figure of Merit (FoM) is a type of assessment of performance as it relates to the function of a system or device. Typically, this figure is represented as a point along a scale, and may involve comparing the merits of the device or system with similar products as a means of arranging that scale. The idea behind the figure of merit is to provide consumers with an idea of the benefits received by choosing one product over another.

Source: https://www.wise-geek.com/what-is-a-figure-of-merit.htm

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:fom

Last update: **2021/06/15 13:14**

File Transfer Protocol (FTP)

Return to Glossary

File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet over TCP/IP connections. FTP is a Client-Server Protocol where a dido:public:ra:xapend:xapend.a_glossary:c:client]] will ask for a file, and a local or remote server will provide it.

The end-users machine is typically called the local host machine, which is connected via the Internet to the remote host—which is the second machine running the FTP software.

Source: https://searchnetworking.techtarget.com/definition/File-Transfer-Protocol-FTP

Financial Instrument Global Identifier (FIGI)

Return to Glossary

The **Financial Instrument Global Identifier (FIGI)** is (formerly Bloomberg Global Identifier (BBGID)) is an open standard, unique identifier of financial instruments that can be assigned to instruments including common stock, options, derivatives, futures, corporate and government bonds, municipals, currencies, and mortgage products

Source: Financial Instrument Global Identifier

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:figi

Last update: **2020/11/13 15:29**

Firewall

Return to Glossary

A **Firewall** is software used to maintain the security of a private network. Firewalls block unauthorized access to or from private networks and are often employed to prevent unauthorized Web users or illicit software from gaining access to private networks connected to the Internet. A firewall may be implemented using hardware, software, or a combination of both.

A firewall is recognized as the first line of defense in securing sensitive information. For better safety, the data can be encrypted.

Source: https://www.techopedia.com/definition/5355/firewall

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:firewall

Last update: **2021/06/15 13:15**

Five Nines

Return to Glossary

Five Nines is the term used for describing the availability of a computer or a service at 99.999 percent of the time it is required. In other words, the system or service is only unavailable for 5.39 minutes throughout the year for planned or unplanned downtime. Five nines is recommended and required for mission-critical requirements and for certain areas such as e-commerce. However, five nines availability has always been a challenge for a service or network and is often impossible to guarantee.

Table 2: Five nines in terms of availability

Number of Nines	percent	Availability Per Year
1	**9%**	332+ days of downtime per year
2	**99%**	3 days, 15 hours and 40 minutes downtime per year
3	**99.9%**	8 hours, 46 minutes downtime per year
4	**99.99%**	52 minutes, 36 seconds downtime per year
5	**99.999%**	5 minutes, 15 seconds or less of downtime in a year

Source: Five Nines

Flowchart

Return to Glossary

A **Flowchart** is a type of diagram representing a process using different symbols containing information about steps or a sequence of events. Each of these symbols is linked with arrows to illustrate the flow direction of the process.

Source: https://www.techopedia.com/definition/5512/flowchart

Full-Disk Encryption (FDE)

Return to Glossary

Full-Disk Encryption (FDE) is encryption at the hardware level. FDE works by automatically converting data on a hard drive into a form that cannot be understood by anyone who doesn’t have the key to “undo” the conversion. Without the proper authentication key, even if the hard drive is removed and placed in another machine, the data remains inaccessible. FDE can be installed on a computing device at the time of manufacturing or it can be added later on by installing a special software driver.

Source: https://whatis.techtarget.com/definition/full-disk-encryption-FDE

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:fde

Last update: **2020/11/15 17:42**

Full Node

Return to Glossary

A **Full Node** is a complete copy of the blockchain and is able to verify all transactions since the beginning. This requires a lot of disk space.

Source: Full Node

Functional Language

Return to Glossary

A **Functional Language** is a programming language built over and around logical functions or procedures within its programming structure. It is based on and is similar to mathematical functions in its program flow.

Functional languages derive their basic structure from the mathematical framework of Lambda calculus and combinatory logic. Erlang, LISP, Haskell and Scala are the most well-known functional languages.

Source: https://www.techopedia.com/definition/19505/functional-language

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:functionallanguage

Last update: **2021/06/15 13:16**

Functional Programming

Return to Glossary

FunctionalProgramming (also called FP) is a way of thinking about software construction by creating pure functions. It avoid concepts of shared state, mutable data observed in Object Oriented Programming.

Functional languages emphasizes on expressions and declarations rather than execution of statements. Therefore, unlike other procedures which depend on a local or global state, value output in FP depends only on the arguments passed to the function.

Source: https://www.guru99.com/functional-programming-tutorial.html

Functional Requirements

Return to Glossary

Functional Requirements define the basic system behaviour. Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs. Functional requirements usually define if/then behaviors and include calculations, data input, and business processes.

Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements.

Also see Non-Functional Requirements.

Source: Functional Requirements

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:funcreq

Last update: **2020/11/13 15:29**

Fungible

Return to Glossary

Fungible (of goods contracted for without an individual specimen being specified) replaceable by another identical item; mutually interchangeable. Example: ‘it is by no means the world’s only fungible commodity’

Source: fungible

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:f:fungible

Last update: **2020/11/13 15:29**

G

Return to Glossary

Create a Glossary entry starting with 'G' **Word or Expression** →	□	Add page

- Gateway
- General Data Protection Regulation (GDPR)
- General-Purpose Graphics Processing Unit (GPGPU)
- Goal
- Google Mobile Services (GMS)
- Government Off-The-Shelf (GOTS)
- Graphical User Interface (GUI)
- Graphics Processing Unit (GPU)
- Greenfield

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:g:start

Last update: **2021/06/14 21:35**

Gateway

Return to Glossary

A **Gateway** normally works at the Transport and Session Layers of the Open Systems Interconnection (OSI) Model. At the Transport layer and above, there are numerous protocols and standards from different vendors; gateways are used to deal with them. Gateways provide translation between networking technologies such as Open System Interconnection (OSI) and Transmission Control Protocol/Internet Protocol (TCP/IP). Because of this, gateways connect two or more autonomous networks, each with its own routing algorithms, protocols, topology, domain name service, and network administration procedures and policies.

Gateways perform all of the functions of routers and more. In fact, a router with added translation functionality is a gateway. The function that does the translation between different network technologies is called a protocol converter.

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:g:gateway

Last update: **2020/11/14 15:56**

General-Purpose Graphics Processing Unit (GPGPU)

Return to Glossary

General-Purpose Graphics Processing Unit (GPGPU) is a graphics processing unit (GPU) that performs non-specialized calculations that would typically be conducted by the Central Processing Unit (CPU) (central processing unit). Ordinarily, the Graphics Processing Unit (GPU) is dedicated to graphics rendering.

GPGPUs are used for tasks that were formerly the domain of high-power CPUs, such as physics calculations, encryption/decryption, scientific computations and the generation of cypto currencies such as Bitcoin.[16]

Source: https://whatis.techtarget.com/definition/GPGPU-general-purpose-graphics-processing-unit

[16] Rouse, Margret; Definition of GPGPU, WhatIs.com, Accessed 8 December 2020, https://whatis.techtarget.com/definition/GPGPU-general-purpose-graphics-processing-unit

General Data Protection Regulation (GDPR)

Return to Glossary

General Data Protection Regulation (GDPR) is a legal framework that sets guidelines for the collection and processing of personal information from individuals who live in the European Union (EU). Since the Regulation applies regardless of where websites are based, it must be heeded by all sites that attract European visitors, even if they don't specifically market goods or services to EU residents.

The GDPR mandates that EU visitors be given a number of data disclosures. The site must also take steps to facilitate such EU consumer rights as a timely notification in the event of personal data being breached. Adopted in April 2016, the Regulation came into full effect in May 2018, after a two-year transition period. Source: General Data Protection Regulation (GDPR)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:g:dgpr

Last update: **2020/11/13 15:29**

Goal

Return to Glossary

A **Goal** is an overarching statement that is meant to provide overall context for an outcome of a project. This outcome is a summary statement of the business value of the project.

Source: Goal

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:g:goal

Last update: **2020/11/13 15:29**

Google Mobile Services (GMS)

Return to Glossary

Google Mobile Services (GMS) are the apps by Google that often come pre-installed on Android devices. GMS is not a part of the Android Open Source Project (AOSP), which means an Android manufacturer needs to obtain a license from Google in order to legally pre-install GMS on an Android device. This license is provided by Google without any license fees.

GMS consists of two parts; a popular bundle package and an other bundle package. In order to gain a license for GMS, the popular bundle package needs to be pre-installed by Android device manufactures, which are usually called pre-loaded apps.

Source: Google Mobile Services (GMS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:g:gms

Last update: **2020/11/13 15:29**

Government Off-The-Shelf (GOTS)

Return to Glossary

Government Off-The-Shelf (GOTS) is a term for software and hardware government products that are ready to use and which were created and are owned by a government agency.

Typically GOTS are developed by the technical staff of the government agency for which it is created. It is sometimes developed by an external entity, but with funding and specification from the agency. Because agencies can directly control all aspects of GOTS products, these are sometimes preferred for government purposes.

GOTS software solutions can normally be shared among government agencies without additional cost.

Source: Government Off-The-Shelf (GOTS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:g:gots

Last update: **2020/11/14 20:21**

Graphical User Interface (GUI)

Return to Glossary

Graphical User Interface (GUI) is a type of user interface through which users interact with electronic devices via visual indicator representations.

Source: Graphical User Interface (GUI)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:g:gui

Last update: **2020/11/13 02:59**

Graphics Processing Unit (GPU)

Return to Glossary

Graphics Processing Unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing. Their highly parallel structure makes them more efficient than general-purpose Central Processing Unit (CPU) for algorithms that process large blocks of data in parallel. In a personal computer, a GPU can be present on a video card or embedded on the motherboard. In certain CPUs, they are embedded on the CPU die.

Source: Graphics Processing Unit (GPU)

Greenfield

Return to Glossary

Greenfield is a type of deployment that refers to the installation of an IT system where previously there was none. This term is derived from the construction industry, where new development on previously undeveloped land is called greenfield development. Greenfield deployment may refer to a network, data center or other major IT projects when they are built from the ground up. This type of development is often beneficial because it is not subject to constraints posed by existing networks.

Greenfield networks may also be referred to as greenfield projects.

Source: Greenfield

H

Return to Glossary

Create a Glossary entry starting with 'H' **Word or Expression** →	▭ Add page

- Hard Fork
- Hardware Firewall
- Health Insurance Portability and Accountability Act (HIPAA)
- History QoS
- Horizontal Scaling
- Hub
- Human-machine interface (HMI)
- Hybrid Network
- Hype-Cycle
- Hypertext Transfer Protocol (HTTP)
- Hypertext transfer protocol (HTTP) Response
- Hypertext transfer protocol (HTTP) Request
- Hypertext Transport Protocol Secure (HTTPS)
- Hypervisor

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:h:start

Last update: **2021/06/15 13:42**

Hard Fork

Return to Glossary

A **hard fork** (or sometimes **hardfork**), as it relates to blockchain technology, is a radical change to the protocol that makes previously invalid blocks/transactions valid (or vice-versa). This requires all nodes or users to upgrade to the latest version of the protocol software.

Source: Hard Fork

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:h:hard_fork

Last update: **2020/11/13 15:29**

Hardware Firewall

Return to Glossary

A **Hardware Firewall** is a physical device similar to a server that filters traffic to a computer. Instead of plugging the network cable into the server, it is connected to the firewall, positioning the firewall between the uplink and the computer. Like a standard computer with a processor, memory, and sophisticated software, these devices also employ powerful networking components (hardware and software) and force all traffic traversing that connection to be inspected by configurable rulesets which grant or deny access accordingly.

Hardware Firewalls use packet filtering to examine the header of a packet to determine its source and destination. This information is compared to a set of predefined or user-created rules that determine whether the packet is to be forwarded or dropped.https://www.webopedia.com/DidYouKnow/Hardware_Software/firewall_types.asp

Source: https://www.liquidweb.com/blog/hardware-firewalls-an-overview-of-benefits-and-how-they-keep-you-secure/

Health Insurance Portability and Accountability Act (HIPAA)

Return to Glossary

Health Insurance Portability and Accountability Act (HIPAA) a US law designed to provide privacy standards to protect patients' medical records and other health information provided to health plans, doctors, hospitals and other health care providers. Developed by the Department of Health and Human Services, these new standards provide patients with access to their medical records and more control over how their personal health information is used and disclosed. They represent a uniform, federal floor of privacy protections for consumers across the country. State laws providing additional protections to consumers are not affected by this new rule. HIPAA took effect on April 14, 2003.

Source: Health Insurance Portability and Accountability Act

History QoS

Return to Glossary

History_qos is …

Source: URI

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:h:history_qos

Last update: **2021/06/15 13:40**

Horizontal Scaling

Return to Glossary

Horizontal Scaling refers to provisioning additional servers to meet your needs, often splitting workloads between servers to limit the number of requests any individual server is getting. In a cloud-based environment, this would mean adding additional instances instead of moving to a larger instance size.

Source: https://cloudcheckr.com/cloud-cost-management/cloud-vs-data-center-what-is-scalability-in-cloud-computing/

Hub

Return to Glossary

A **Hub** connects multiple computer networking devices together. A hub also acts as a repeater in that it amplifies signals that deteriorate after traveling long distances over connecting cables. A hub is the simplest in the family of network connecting devices because it connects LAN components with identical protocols.

A **Hub** can be used with both digital and analog data, provided its settings have been configured to prepare for the formatting of the incoming data. For example, if the incoming data is in digital format, the hub must pass it on as packets; however, if the incoming data is analog, then the hub passes it on in signal form.

Hubs do not perform packet filtering or addressing functions; they just send data packets to all connected devices. Hubs operate at the Physical layer of the Open Systems Interconnection (OSI) Model. There are two types of hubs: simple and multiple port.

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

Human-machine interface (HMI)

Return to Glossary

Human-machine interface (HMI) is a component of certain devices that are capable of handling human-machine interactions. The interface consists of hardware and software that allow user inputs to be translated as signals for machines that, in turn, provide the required result to the user. Human-machine interface technology has been used in different industries like electronics, entertainment, military, medical, etc. Human-machine interfaces help in integrating humans into complex technological systems.

- **NOTE:** Human-machine interface is also known as man-machine interface (MMI), computer-human interface or human-computer interface.

Source: Human-machine interface (HMI)

Hybrid Network

Return to Glossary

Hybrid Blockchain is unique in that it is decentralized while also making it possible to restrict the visibility of information on the network with a combination of Public, Private, Permissionless and Permissioned Networks. Thus, a hybrid blockchain is appealing for regulated markets as it offers the benefits of public blockchain and private blockchain together.[17]

[17] “Hybrid Blockchain: Decentralized Option for Highly Regulated Markets - Few players in highly regulated markets have adopted blockchain technology. However, hybrid blockchain will change this.”, Mina Down, 14 November 2018, Source: https://blog.goodaudience.com/hybrid-blockchain-decentralize-highly-regulated-markets-900f30a37903

Hype-Cycle

Return to Glossary

The **hype-cycle** is a graphical representation of the life cycle stages a technology goes through from conception to maturity and widespread adoption.

Source: Gartner hype cycle

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:h:hype_cycle

Last update: **2020/11/13 15:29**

Hypertext Transfer Protocol (HTTP)

Return to Glossary

Hypertext Transfer Protocol (HTTP) is a fundamental Protocol used on the Internet in order to control data transfer to and from a hosting Server, in communication with a web browser Client.

HTTP is the essential means of communication between web users and the servers or services that maintain the websites themselves.

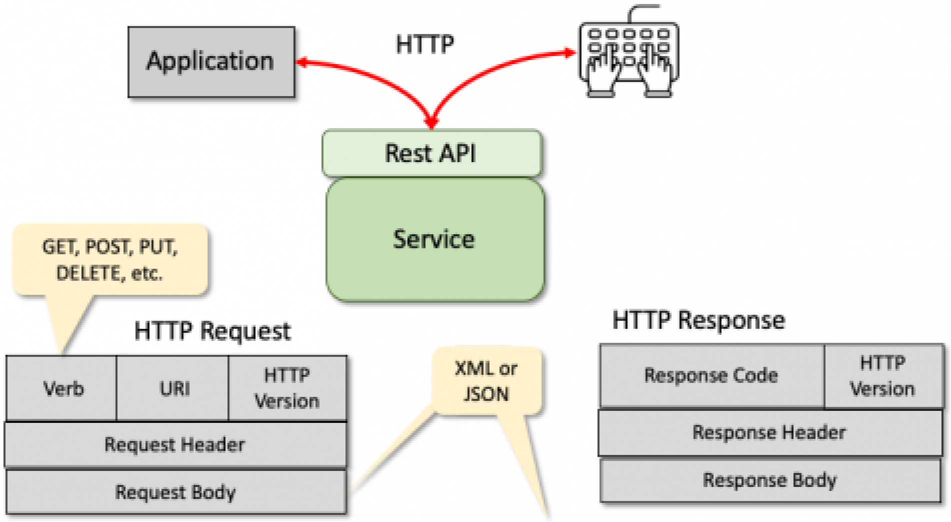


Figure 5: The HTTP Request/Response Model

See:

- Hypertext transfer protocol (HTTP) Request
- Hypertext transfer protocol (HTTP) Response

See: RFC7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication

Source: https://www.techopedia.com/definition/2336/hypertext-transfer-protocol-http

Hypertext transfer protocol (HTTP) Request

Return to Glossary

Hypertext transfer protocol (HTTP) Request is the message that is sent by a client to a server or service over Hypertext Transfer Protocol (HTTP). When these requests are being sent, clients can use various methods.

Source: https://rapidapi.com/blog/api-glossary/http-request-methods

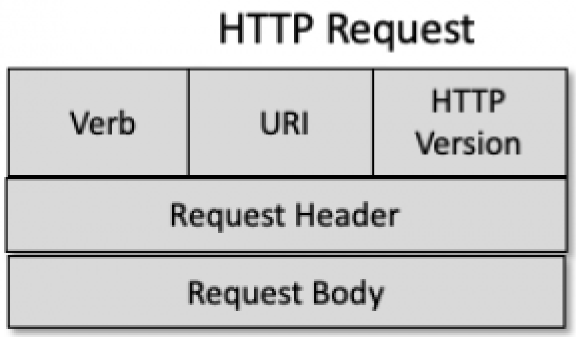


Figure 6: HTTP Request

An HTTP Request has five major parts −

- **Verb** − Indicates the HTTP methods such as GET, POST, DELETE, PUT, etc.
- **URI** − Uniform Resource Identifier (URI) to identify the resource on the server.
- **HTTP Version** − Indicates the HTTP version. For example, HTTP v1.1.
- **Request Header** − Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by the client, format of the message body, cache settings, etc.
- **Request Body** − Message content or Resource representation (usually XML or JSON)

Source: https://www.tutorialspoint.com/restful/restful_messages.htm

Hypertext transfer protocol (HTTP) Response

Return to Glossary

HTTP Response occurs on Hypertext Transfer Protocol (HTTP) after receiving and interpreting a Hypertext transfer protocol (HTTP) Request message, a Server or Service responds with a response.

Source: https://www.tutorialspoint.com/http/http_responses.htm

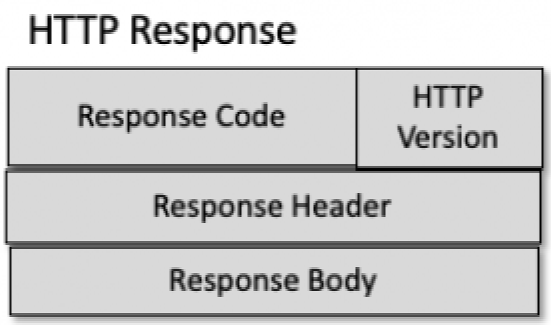


Figure 7: The HTTP Response

An HTTP Response has four major parts –

- **Status/Response Code** – Indicates the Server status for the requested resource. For example, 404 means resource not found and 200 means response is ok.
- **HTTP Version** – Indicates the HTTP version. For example HTTP v1.1.
- **Response Header** – Contains metadata for the HTTP Response message as key value pairs. For example, content length, content type, response date, server type, etc.
- **Response Body** – Response message content or Resource representation.

Source: https://www.tutorialspoint.com/restful/restful_messages.htm

Hypertext Transport Protocol Secure (HTTPS)

Return to Glossary

Hypertext Transfer Protocol Secure (HTTPS) is a variant of the standard web Hypertext Transfer Protocol (HTTP) that adds a layer of security on the data in transit through a Secure Sockets Layer (SSL) or Transport layer security (TLS) Protocol connection.

HTTPS enables encrypted communication and secure connection between a remote user (i.e., Client and the primary web Server.

See: RFC2818 - HTTP Over TLS (HTTPS)

Source https://www.techopedia.com/definition/5361/hypertext-transport-protocol-secure-https

Hypervisor

Return to Glossary

A **Hypervisor**, also known as a **Virtual Machine Monitor (VMM), is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing. Source: https://www.vmware.com/topics/glossary/content/hypervisor ~~DISCUSSION:on|Outstanding Issues~~ ~~DISCUSSION:off~~**

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:h:hypervisor

Last update: **2021/06/15 15:36**

I

Return to Glossary

Create a Glossary entry starting with 'I' **Word or Expression** →		Add page

- Identification
- Identifier (ID)
- Immutable
- Independent Event
- Industrial Internet of Things (IIoT)
- Information Assurance (IA)
- Information Security (IS/InfoSec)
- Information Technology (IT)
- Infrared Wireless Networking
- Infrastructure-as-a-Service (IaaS)
- Initial Coin Offering (ICO)
- Installability
- Instance
- Institute of Electrical and Electronics Engineers (IEEE)
- Integration Testing
- Intellectual Property (IP)
- Interface
- Interface Testing
- International Organization for Standardization (ISO)
- Internet
- Internet of Things (IOT)
- Internet Protocol (IP)
- Internet Protocol address (IP address)
- Internet Service Provider (ISP)
- Interoperability
- Interoperability Testing
- ISO 15288
- ISO 90003-2018
- ISO 9001

Identification

Return to Glossary

Identification is the starting point for all access control as without proper identification it will not be possible to grant resources to any identity. The main objective of identification is to bind a user to appropriate controls based on the identity.

Source: Identification

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:identification

Last update: **2020/11/13 15:29**

Identifier (ID)

Return to Glossary

Identifiers (IDs) are symbols used to uniquely identify a program element in the code. They are also used to refer to types, constants, macros and parameters. An identifier name should indicate the meaning and usage of the element being referred.

Source: ID (Techopedia)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:id

Last update: **2021/06/15 13:43**

Immutable

Return to Glossary

Immutable is an unchanging over time or unable to be changed[18]. The immutability property of a blockchain refers to the fact that any data once written on the blockchain cannot be changed. [19]

[18] “Immutable”, Lexico, Oxford English Dictionary, 28 June 2018,

[19] “Blockchain Technology Explained: Introduction, Meaning, and Applications”, Mayank Pratap, 1 August 2018, Hackernoon, https://hackernoon.com/blockchain-technology-explained-introduction-meaning-and-applications-edbd6759a2b2

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:immutable

Last update: **2020/11/13 15:29**

Independent Event

Return to Glossary

Independent Event is when the probability of an event is not affected by a previous event.

Source: Independent Event

Industrial Internet of Things (IIoT)

Return to Glossary

Industrial Internet of Things (IIoT) is a term for all of the various sets of hardware pieces that work together through internet of things connectivity to help enhance manufacturing and industrial processes. When people talk about the industrial internet of things, they're talking about all of the sensors, devices and machines that contribute to physical business processes in industrial settings. By contrast, when people talk about the internet of things in general, they're talking about any connected devices that fit the IoT model – for instance, when people think about the internet of things, they often think about smart home devices that are linked together to provide consumer conveniences. Source: Industrial Internet of Things (IIoT)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:iiot

Last update: **2020/11/13 15:29**

Information Assurance (IA)

Return to Glossary

Information Assurance (IA) are measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. These measures include providing for restoration of information systems by incorporating protection, detection, and reaction capabilities

Source: Committee on National Security Systems, CNS 4009 Glossary

Source: NYSE Assurance

Information Security (IS/InfoSec)

Return to Glossary

Information security, often referred to as **InfoSec**, is designed to protect the confidentiality, integrity and availability of computer system data from those with malicious intentions. Confidentiality, integrity and availability are sometimes referred to as the CIA Triad of information security. This triad has evolved into what is commonly termed the Parkerian hexad, which includes confidentiality, possession (or control), integrity, authenticity, availability and utility

Source: Information Security (IS/InfoSec)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:is

Last update: **2020/11/13 15:29**

Information Technology (IT)

Return to Glossary

Information Technology (IT) is a business sector that deals with computing, including hardware, software, telecommunications and generally anything involved in the transmittal of information or the systems that faciliate communication.

Source: [[https://www.techopedia.com/definition/626/information-technology-it | Information Technology (IT)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:infotech

Last update: **2020/11/14 13:26**

Infrared Wireless Networking

Return to Glossary

Infrared Wireless Networking uses infrared beams to send data transmissions between devices. Infrared wireless networking offers higher transmission rates, reaching 10Mbps to 16Mbps.

As expected, infrared light beams cannot penetrate objects; therefore, the signal is disrupted when something blocks the light. Infrared can be either a directed (line-of-sight) or diffuse technology. A directed infrared system provides a limited range of approximately 3 feet and typically is used for personal area networks. Diffused infrared can travel farther and is more difficult to block with a signal object. Diffused infrared wireless LAN systems do not require line of sight, but usable distance is limited to room distances.

Infrared provides a secure, low-cost, convenient cable-replacement technology. It is well suited for many specific applications and environments.

Source: Infrared Wireless Networking

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:infrared

Last update: **2020/11/16 19:00**

Infrastructure-as-a-Service (IaaS)

Return to Glossary

Iaas is …Infrastructure-as-a-Service (IaaS), also known as cloud infrastructure services, is a form of cloud computing in which IT infrastructure is provided to end users through the internet. IaaS is commonly associated with serverless computing.

Source: Infrastructure-as-a-Service (IaaS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:iaas

Last update: **2020/11/13 15:29**

Initial Coin Offering (ICO)

Return to Glossary

Initial Coin Offering (ICO) is the cryptocurrency industry’s equivalent to an Initial Public Offering (IPO). ICOs act as a way to raise funds, where a company looking to raise money to create a new coin, app, or service launches an ICO. Interested investors can buy into the offering and receive a new cryptocurrency token issued by the company. This token may have some utility in using the product or service the company is offering, or it may just represent a stake in the company or project.

Source: Initial Coin Offering (ICO)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:ico

Last update: **2020/11/13 15:29**

Installability

Return to Glossary

See 2.2.1.2 Installability

Installability is capability of the software product to be installed in a specified environment.

Source: ISO 9126: 2001, 6.6.2

Instance

Return to Glossary

An **Instance** is a unique element within a Topic, described by unique values of key fields.

Instances represent individual real-world objects within a single data stream. For example, you can represent individual radar tracks within a single “TrackData” topic, or individual components of a wind turbine within a “TurbineComponentState” topic. What constitutes an instance will be different depending on what you are representing in your Topic.

Some examples of instances you can represent in data:

- Unique vehicles in a “Vehicle Position” topic
- Unique applications in an “Application State” topic
- Unique temperature sensors in a “Temperature” topic

Modeling your data with unique instances gives you several advantages, including:

- Scalability when representing many thousands of objects - you can use a single DataReader to receive the state of all the instances within a topic
- Quality of Service applied per-instance: Some quality of service behaviors are applied per-instance. For example, the HISTORY QoS is applied per-instance, which allows you to store a certain number of updates for every instance. Note that you cannot specify different QoS for specific instances within the same DataWriter or DataReader. The QoS is set on the writer or reader, and applied on each instance. A description of all QoS, including information on which QoS policies are applied per-instance is here.
- Life-cycle information is available for instances: A receiving application can detect whether an instance is alive or not. A sending application can change the instance state.
- Efficient filtering of data by key fields: If you are only interested in receiving updates about certain instances, you can set up a content-filter for certain values of the key fields. This filtering is particularly efficient if you are only filtering on key fields.

Source: https://community.rti.com/glossary/instance

Institute of Electrical and Electronics Engineers (IEEE)

Return to Glossary

IEEE is a global association and organization of professionals working toward the development, implementation and maintenance of technology-centered products and services.

IEEE is a nonprofit organization founded in 1963. It works solely toward innovating, educating and standardizing the electrical and electronic development industry. It is best known for its development of standards such as IEEE 802.11.

Source: IEEE (Techopedia)

Integration Testing

Return to Glossary

Integration Testing is performed to test individual components to check how they function together. In other words, it is performed to test the modules which are working fine individually and do not show bugs when integrated. It is the most common functional testing type and performed as automated testing.

Generally, developers build different modules of the system/software simultaneously and don't focus on others. They perform extensive black box and white box functional verification, commonly known as unit tests, on the individual modules. Integration tests cause data and operational commands to flow between modules which means that they have to act as parts of a whole system rather than individual components. This typically uncovers issues with UI operations, data formats, operation timing, API calls, and database access and user interface operation.

Source: https://www.simform.com/functional-testing-types/

Intellectual Property (IP)

Return to Glossary

Intellectual Property (IP) is any product of the human intellect that the law protects from unauthorized use by others. The ownership of intellectual property inherently creates a limited monopoly in the protected property. Intellectual property is traditionally comprised of four categories: patent, copyright, trademark, and trade secrets. Source: Intellectual Property (IP)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:intelp

Last update: **2020/11/13 15:29**

Interface

Return to Glossary

Interface is a boundary across which two independent systems meet and act on or communicate with each other. In computer technology, there are several types of interfaces.

Source: Interface

Interface Testing

Return to Glossary

Interface Testing is defined as a software testing type which verifies whether the communication between two different software systems is done correctly.

A connection that integrates two components is called interface. This interface in a computer world could be anything like Application Programming Interface (API), web services, etc. Testing of these connecting services or interface is referred to as Interface Testing.

An interface is actually software that consists of sets of commands, messages, and other attributes that enable communication between a device and a user.

Source: https://www.guru99.com/interface-testing.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:interfacetesting

Last update: **2020/12/20 21:04**

International Organization for Standardization (ISO)

Return to Glossary

International Organization for Standardization (ISO) is an international standard-setting body composed of representatives from various national standards organizations. In contrast to many international organizations, which utilize the British English form of spelling, the ISO uses English with Oxford spelling as one of its official languages along with French and Russian.

Founded on 23 February 1947, the organization promotes worldwide proprietary, industrial, and commercial standards. It is headquartered in Geneva, Switzerland, and works in 164 countries.

It was one of the first organizations granted general consultative status with the United Nations Economic and Social Council.

Source: International Organization for Standardization (ISO)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:iso

Last update: **2020/11/14 12:16**

Internet

Return to Glossary

The **Internet** is the global system of interconnected computer networks that uses the Internet protocol suite (Transmission Control Protocol (TCP)/Internet Protocol (IP)) to communicate between networks and devices. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet carries a vast range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web (WWW), electronic mail, telephony, and file sharing.

Source: https://en.wikipedia.org/wiki/Internet

Internet of Things (IOT)

Return to Glossary

Internet of Things (IOT) is a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.

NOTE 1

> Through the exploitation of identification, data capture, processing and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled.

NOTE 2

> From a broader perspective, the IoT can be perceived as a vision with technological and societal

implications.

Source: ITU-T Y.2060 - Overview of the Internet of things

Internet Protocol (IP)

Return to Glossary

Internet Protocol (IP) is the principal set (or communications protocol) of digital message formats and rules for exchanging messages between computers across a single network or a series of interconnected networks, using the Internet Protocol Suite (often referred to as TCP/IP). Messages are exchanged as datagrams, also known as data packets or just packets.

IP is the primary protocol in the Internet Layer of the Internet Protocol Suite, which is a set of communications protocols consisting of four abstraction layers: link layer (lowest), Internet layer, transport layer and application layer (highest).

The main purpose and task of IP is the delivery of datagrams from the source host (source computer) to the destination host (receiving computer) based on their addresses. To achieve this, IP includes methods and structures for putting tags (address information, which is part of metadata) within datagrams.

The process of putting these tags on datagrams is called encapsulation.

Source: Internet Protocol (IP)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:ip

Last update: **2020/11/13 15:29**

Internet Protocol address (IP address)

Return to Glossary

An **Internet Protocol address (IP address)** is a logical numeric address that is assigned to every single computer, printer, switch, router or any other device that is part of a TCP/IP-based network.

The Internet Protocol (IP) address is the core component on which the networking architecture is built; no network exists without it. An IP address is a logical address that is used to uniquely identify every node in the network. Because IP addresses are logical, they can change. They are similar to addresses in a town or city because the IP address gives the network node an address so that it can communicate with other nodes or networks, just like mail is sent to friends and relatives.

The numerals in an IP address are divided into 2 parts:

- The network part specifies which networks this address belongs to and
- The host part further pinpoints the exact location.

Source: Internet Protocol address (IP address)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:ipaddr

Last update: **2020/11/14 14:20**

Internet Service Provider (ISP)

Return to Glossary

An **Internet Service Provider (ISP)** is a company that provides customers with Internet access. Data may be transmitted using several technologies, including dial-up, DSL, cable modem, wireless or dedicated high-speed interconnects.

Typically, ISPs also provide their customers with the ability to communicate with one another by providing Internet email accounts, usually with numerous email addresses at the customer's discretion. Other services, such as telephone and television services, may be provided as well. The services and service combinations may be unique to each ISP.

An Internet service provider is also known as an Internet Access Provider (IAP).

Source: https://www.techopedia.com/definition/2510/internet-service-provider-isp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:isp

Last update: **2020/11/16 18:50**

Interoperability

Return to Glossary

See 2.2.6 Interoperability

Interoperability is a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, in either implementation or access, without any restrictions.

Source: Interoperability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:interoperability

Last update: **2021/06/15 13:44**

Interoperability Testing

Return to Glossary

Interoperability Testing is a software testing type, that checks whether the software can interact with other software components and systems. The purpose of Interoperability tests is to ensure that the software product is able to communicate with other components or devices without any compatibility issues.

In other words, interoperability testing means to prove that end-to-end functionality between two communicating systems is as specified by the requirements. For example, interoperability testing is done between smartphones and tablets to check data transfer via Bluetooth. Source: https://www.guru99.com/interoperability-testing.html#:~:text=INTEROPERABILITY%20TESTING%20is%20a%20software,devices%20without%20any%20compatibility%20issues.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:i:interoptesting

Last update: **2021/06/15 13:48**

ISO 9001

Return to Glossary

ISO 9001 sets out the criteria for a quality management system and is the only standard in the family that can be certified to (although this is not a requirement). It can be used by any organization, large or small, regardless of its field of activity. In fact, there are over one million companies and organizations in over 170 countries certified to ISO 9001.

This standard is based on a number of quality management principles including a strong customer focus, the motivation and implication of top management, the process approach and continual improvement. These principles are explained in more detail in ISO’s quality management principles. Using ISO 9001 helps ensure that customers get consistent, good-quality products and services, which in turn brings many business benefits.

See Aldo: ISO 9001:2015 Quality management

Source: https://www.iso.org/iso-9001-quality-management.html

ISO 15288

Return to Glossary

ISO/IEC/IEEE 15288:2015 establishes a common framework of process descriptions for describing the life cycle of systems created by humans. It defines a set of processes and associated terminology from an engineering viewpoint. These processes can be applied at any level in the hierarchy of a system's structure. Selected sets of these processes can be applied throughout the life cycle for managing and performing the stages of a system's life cycle. This is accomplished through the involvement of all stakeholders, with the ultimate goal of achieving customer satisfaction.

ISO/IEC/IEEE 15288:2015 also provides processes that support the definition, control and improvement of the system life cycle processes used within an organization or a project. Organizations and projects can use these processes when acquiring and supplying systems.

ISO/IEC/IEEE 15288:2015 concerns those systems that are man-made and may be configured with one or more of the following system elements: hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, materials and naturally occurring entities.

See also: ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes

Source: https://www.iso.org/standard/63711.html

ISO 90003-2018

Return to Glossary

90003-2018 - ISO/IEC/IEEE International Standard - Software engineering – Guidelines for the application of ISO 9001:2015 to computer software covering the System Lifecycle phases of acquisition, supply, development, operation and maintenance of computer software and related support services.

Source: https://standards.ieee.org/standard/90003-2018.html

J

Return to Glossary

Create a Glossary entry starting with 'J' **Word or Expression** →	□ Add page

- Java Database Connectivity(JDBC)
- Javascript
- JavaScript Object Notation (JSON)
- Jitter
- Just-In-Time (JIT)

=-

Java Database Connectivity(JDBC)

Return to Glossary

Jdbc is … **Java Database Connectivity(JDBC)** is a Java Application Programming Interface (API) that enables Java programs to execute Structured Query Language (SQL) statements. This allows Java programs to interact with any SQL-compliant database. Since nearly all Relational DataBase Management Systems (RDBMSs) support SQL, and because Java itself runs on most platforms, JDBC makes it possible to write a single database application that can run on different platforms and interact with different DBMSs.

JDBC is similar to Open Database Connectivity (ODBC), but is designed specifically for Java programs, whereas ODBC is language-independent.

JDBC was developed by JavaSoft, a subsidiary of Sun Microsystems.

Source: https://www.webopedia.com/definitions/jdbc/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:j:jdbc

Last update: **2021/06/15 13:48**

Javascript

Return to Glossary

JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time (JiT) compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Source: https://en.wikipedia.org/wiki/JavaScript

JavaScript Object Notation (JSON)

Return to Glossary

Javascript Object Notation (JSON) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values). It is a very common data format, with a diverse range of applications, one example being web applications that communicate with a server.

JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. JSON filenames use the extension `.json`.

Source: https://en.wikipedia.org/wiki/JSON

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:j:json

Last update: **2021/06/15 13:49**

Jitter

Return to Glossary

Jitter is the variation in the time between data packets arriving, caused by network congestion, or route changes. … The standard jitter measurement is in milliseconds (ms). If receiving jitter is higher than 15-20ms, it can increase latency and result in packet loss, causing audio quality degradation

Source: Jitter

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:j:jitter

Last update: **2020/11/16 20:56**

Just-In-Time (JIT)

Return to Glossary

Just-In-Time (JIT) compiler is a an essential part of the JRE i.e. Java Runtime Environment, that is responsible for performance optimization of java based applications at run time. Compiler is one of the key aspects in deciding performance of an application for both parties i.e. the end user and the application developer.

Source: Just-In-Time (JIT)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:j:jit

Last update: **2020/11/13 15:29**

K

Return to Glossary

Create a Glossary entry starting with 'D' **Word or Expression** →		Add page

- Key
- Know Your Customer (KYC)

=-
=-

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:k:start

Last update: **2021/06/14 21:36**

Key

Return to Glossary

A **Key** is one or more fields within the data type. A key uniquely identifies one instance of a Topic from another instance of the same Topic.

Source: Key

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:k:key

Last update: **2020/11/14 18:48**

Know Your Customer (KYC)

Return to Glossary

Know Your Client or Know Your Customer (KYC) is a standard in the investment industry that ensures investment advisors know detailed information about their clients' risk tolerance, investment knowledge, and financial position. KYC protects both clients and investment advisors. Clients are protected by having their investment advisor know what investments best suit their personal situations. Investment advisors are protected by knowing what they can and cannot include in their client's portfolio. KYC compliance typically involves requirements and policies such as risk management, customer acceptance policies, and transaction monitoring. Source: Know Your Customer (KYC)

L

Return to Glossary

Create a Glossary entry starting with 'L' **Word or Expression** →	□	Add page

- Latency
- Learnability
- Ledger
- License Distribution
- License Linking
- License Modification
- License Patent Grant
- License Private Use
- Licensing Sublicensing
- Licensing Trademark Grant
- Light Node
- Lightning Network
- Listener
- Little-Endian
- Liveliness
- Local Area Network (LAN)
- Logical Integrity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:start

Last update: **2021/06/14 21:36**

Latency

Return to Glossary

Latency is a networking term to describe the total time it takes a data packet to travel from one node to another. In other contexts, when a data packet is transmitted and returned back to its source, the total time for the round trip is known as latency. Latency refers to time interval or delay when a system component is waiting for another system component to do something. This duration of time is called **latency**.

Source: Latency

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:latency

Last update: **2020/11/13 15:29**

Learnability

Return to Glossary

Learnability is degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:learnability

Last update: **2021/06/15 13:49**

Ledger

Return to Glossary

Ledger – is a collection of an entire group of similar accounts in double-entry bookkeeping. Also, called book of final entry, a ledger records classified and summarized financial information from journals (the 'books of first entry') as debits and credits, and shows their current balances. In manual accounting systems, a ledger is usually a loose-leaf binder with a separate page for each ledger account. In computerized systems, it consists of interlinked digital files, but follows the same accounting principles as the manual system.

Source: Ledger

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:ledger

Last update: **2020/11/13 15:29**

License Distribution

Return to Glossary

License Distribution is distribution of the code to third parties.

Source: Distribution

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:lic_distribution

Last update: **2020/11/13 15:29**

License Linking

Return to Glossary

License Linking is - linking of the licensed code with code licensed under a different license (e.g. when the code is provided as a library)

Source: Linking

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:lic_linking

Last update: **2020/11/13 15:29**

License Modification

Return to Glossary

License Modification is modification of the code by a licensee.

Source: Modification

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:lic_modification

Last update: **2020/11/13 15:29**

License Patent Grant

Return to Glossary

License Patent Grant is protection of licensees from patent claims made by code contributors regarding their contribution, and protection of contributors from patent claims made by licensees.

Source: Patent Grant

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:lic_patent_grant

Last update: **2020/11/13 15:29**

License Private Use

Return to Glossary

License Private Use is whether modification to the code must be shared with the community or may be used privately (e.g. internal use by a corporation)

Source: Private Use

Licensing Sublicensing

Return to Glossary

Licensing Sublicensing is whether modified code may be licensed under a different license (for example a copyright) or must retain the same license under which it was provided.

Source: Sublicensing

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:lic_sublicensing

Last update: **2020/11/13 15:29**

Licensing Trademark Grant

Return to Glossary

Licensing Trademark Grant is use of trademarks associated with the licensed code or its contributors by a licensee.

Source: Trademark Grant

Lightning Network

Return to Glossary

A **Lightning Network** is network layered on top of an existing blockchain technology (i.e., Bitcoin) intended to help with scalability issues. It is a decentralized system for instant, high-volume micropayments that removes the risk of delegating custody of funds to trusted third parties using a peer-to-peer (P2P) network connections called Micropayment Channel or Payment Channel. [32]

The Lightning Network solves these problems. It is one of the first implementations of a multi-party Smart Contract (programmable money) using bitcoin's built-in scripting. The Lightning Network is leading technological development in multiparty financial computations with bitcoin.

Instant Payments:

Bitcoin aggregates transactions into blocks spaced ten minutes apart. Payments are widely regarded as secure on bitcoin after confirmation of six blocks, or about one hour. On the Lightning Network, payments don't need block confirmations, and are instant and atomic. Lightning can be used at retail point-of-sale terminals, with user device to-device transactions, or anywhere instant payments are needed.

Micropayments:

New markets can be opened with the possibility of micropayments. Lightning enables one to send funds down to 0.00000001 bitcoin without custodial risk. The bitcoin blockchain currently enforces a minimum output size many hundreds of times higher, and a fixed per-transaction fee which makes micropayments impractical. Lightning allows minimal payments denominated in bitcoin, using actual bitcoin transactions.

Scalability:

The bitcoin network will need to support orders of magnitude higher transaction volume to meet demand from automated payments. The coming increase in internet-connected devices needs a platform for machine-to-machine payments and automated micropayment services. Lightning Network transactions are conducted to the blockchain without delegation of trust and ownership, allowing users to conduct nearly unlimited transactions between other devices.

How it Works?

Funds are placed into a two-party, multisignature “channel” bitcoin address. This channel is

> represented as an entry on the bitcoin public ledger. In order to spend funds from the channel, both parties must agree on the new balance. The current balance is stored as the most recent transaction signed by both parties, spending from the channel address. To make a payment, both parties sign a new exit transaction spending from the channel address. All old exit transactions are invalidated by doing so.

The Lightning Network does not require cooperation from the counterparty to exit the channel. Both parties have the option to unilaterally close the channel, ending their relationship. Since all parties have multiple multisignature channels with many different users on this network, one can send a payment to any other party across this network.

[32] “The Bitcoin Lightning Network”, Lightning Network

Light Node

Return to Glossary

A **Light Node** does not store the Tangle and you don't need any neigbours. You just need to select a host (e.g. `https://node.tangle.works:443`) which is a full node itself. This host provides your wallet with all the necessary information to make transaction.

Source: Light Node

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:light

Last update: **2020/11/13 15:29**

Listener

Return to Glossary

Provides a generic mechanism for the middleware to notify an application of status changes in the Entity to which it is attached

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:listener

Last update: **2021/06/15 13:50**

Little-Endian

Return to Glossary

Little-Endian is a type of addressing that refers to the order of data stored in memory. In this convention, the least significant bit (or “littlest” end) is first stored at address 0, and subsequent bits are stored incrementally.

- **Note:** Little-endian is the opposite of big-Big-Endian

Source: https://www.techopedia.com/definition/12892/little-endian

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:littleendian

Last update: **2021/06/15 13:51**

Liveliness

Return to Glossary

Liveliness is …

Source: URI

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:liveliness

Last update: **2021/06/15 13:51**

Local Area Network (LAN)

Return to Glossary

A **Local Area Network (LAN)**is a computer network within a small geographical area such as a home, school, computer laboratory, office building or group of buildings.

A LAN is composed of inter-connected workstations and personal computers which are each capable of accessing and sharing data and devices, such as printers, scanners and data storage devices, anywhere on the LAN. LANs are characterized by higher communication and data transfer rates and the lack of any need for leased communication lines.

Source: Local Area Network (LAN)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:lan

Last update: **2020/11/13 02:58**

Logical Integrity

Return to Glossary

Logical Integrity keeps data unchanged as it’s used in different ways in a relational database. Logical integrity protects data from human error and hackers as well, but in a much different way than physical integrity does. There are four types of logical integrity.

Source: Logical Integrity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:l:logicalintegrity

Last update: **2020/11/15 21:06**

M

Return to Glossary

Create a Glossary entry starting with 'M' **Word or Expression** →		Add page

- Machine Learning (ML)
- Maintainability
- Maintainability Measure
- Manageability
- Management Level
- Manufacturing Execution System (MES)
- Massively Parallel Processing (MPP)
- Mean Active Maintenance Down Time (MAMDT)
- Mean Logistics Delay Time (MLDT)
- Mean Time Between Failure (MTBF)
- Mean Time To Failure (MTTF)
- Mean Time To Repair (MTTR)
- Media Access Control (MAC)
- Message Queue(MQ)
- Message-Oriented Middleware (MOM)
- Metadata
- Microcontroller
- Micropayment Channel
- Middleware
- Miner Node
- Mission Assurance (MA)
- Mission Critical System
- Modem
- Modifiability
- Modified Off-The-Shelf (MOTS)
- Modularity
- Module
- Monitoring Software
- Motherboard
- Multi-core Processor
- Multi-Signature (multisig)
- Must (Requirement)

Maintainability

Return to Glossary

See 2.2.3 Maintainability

Maintainability refers to the ease with which maintenance activities can be performed on an asset or equipment. Its purpose is to measure the probability that a piece of equipment in a failed state can be restored to normal operating conditions after undergoing maintenance.

Source: Maintainability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:maintainability

Last update: **2020/11/13 15:29**

Maintainability Measure

Return to Glossary

Maintainability represents the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers (ISO 25010). Maintainability incorporates such concepts as changeability, modularity, understandability, testability, and reusability. Maintainability is responding rapidly to market conditions and keeping IT costs under control. The Maintainability of an application is a combination of compliance with good coding practices, the homogeneity with which coding rules are applied across an application, and compliance with architectural rules.

Source: IETF - The OAuth 2.0 Authorization Framework - October 2012

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:maintainability_measure

Last update: **2020/11/13 15:29**

Manageability

Return to Glossary

See: 4.2.7 Manageability

Manageability is an indicator of the difficulty or ease of managing a system, a component or a collection of components when changes occur to correct faults, improve performance, or adapt to changes in the computational ecosystem where the system exists. Manageability is also closely tied to the tools used to monitor or maintain the system. The more tools and the better the tools are integrated into a unified user experience, the more manageable is the system.

Source: Manageability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:manageability

Last update: **2021/06/15 13:52**

Management Level

Return to Glossary

Management Level is fifth and highest level of the Automation Pyramid. It uses the companies integrated management system such as as Enterprise Resource Planning (ERP). Corporate management visualize and control operations. This level allows the businesses monitor all levels (i.e., manufacturing, to sales, to purchasing, to finance and payroll). The integration of an ERP promotes efficiency and transparency within a company by helping to communicate the levels.

Source: Management Level

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:manaelevel

Last update: **2020/11/15 20:27**

Manufacturing Execution System (MES)

Return to Glossary

A **Manufacturing Execution System (MES)** is a software control system for managing work processes in industrial situations. Businesses can use this software as part of an enterprise resource planning solution for tracking manufacturing data in real time.

Source: Manufacturing Execution System (MES)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:mes

Last update: **2020/11/15 20:23**

Massively Parallel Processing (MPP)

Return to Glossary

Massively Parallel Processing (MPP) is the coordinated processing of a program by multiple processor s that work on different parts of the program, with each processor using its own operating system and memory . Typically, MPP processors communicate using some messaging interface. In some implementations, up to 200 or more processors can work on the same application. An “interconnect” arrangement of data paths allows messages to be sent between processors. Typically, the setup for MPP is more complicated, requiring thought about how to partition a common database among processors and how to assign work among the processors. An MPP system is also known as a “loosely coupled” or “shared nothing” system.

An MPP system is considered better than a symmetrically parallel system (SMP) for applications that allow a number of databases to be searched in parallel. These include decision support system and data warehouse applications.[21)]

Source: https://whatis.techtarget.com/definition/MPP-massively-parallel-processing

[21)] Rouse, Margret; Definition of massively parallel processing, WhatIs.com, Accessed 8 December 2020, https://whatis.techtarget.com/definition/MPP-massively-parallel-processing

Mean Active Maintenance Down Time (MAMDT)

Return to Glossary

Mean Active Maintenance Down Time (MAMDT) is associated with Planned Maintenance System support philosophy. This is average amount of time while the system is not 100% operational because of diagnostic testing that requires down time.

For example, an automobile that requires 1 day of maintenance every 90 days has a Mean Active Maintenance Down Time of just over 1%.

This is separate from the type of down time associated with repair activities.

Source: Mean Active Maintenance Down Time (MAMDT)

Mean Logistics Delay Time (MLDT)

Return to Glossary

Mean Logistics Delay Time (MLDT) is the average time required to obtain replacement parts from the manufacturer and transport those parts to the work site.

Source: Mean Logistics Delay Time (MLDT)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:mldt

Last update: **2020/11/14 13:35**

Mean Time Between Failure (MTBF)

Return to Glossary

Mean Time Between Failure (MTBF) depends upon the maintenance philosophy. MTBF is the predicted elapsed time between inherent failures of a mechanical or electronic system, during normal system operation. MTBF can be calculated as the arithmetic mean (average) time between failures of a system. The term is used for repairable systems, while Mean Time To Failure (MTTF) denotes the expected time to failure for a non-repairable system.[22]

If a system is designed with both redundancy and automatic fault bypass, then MTBF is the anticipated lifespan of the system if these features cover all possible failure modes (infinity for all practical purposes). Such systems will continue without noticeable interruption when these conditions are satisfied unless there are secondary failures. This is called active redundancy, which requires no maintenance to prevent mission failure. Active redundancy is required for systems that cannot be maintained, such as satellites.

If a system has no redundancy, then MTBF is the inverse of failure rate, λ .

Systems with spare parts that are energized but that lack automatic fault bypass are not actually redundant because human action is required to restore operation after every failure. This depends upon Condition-based maintenance and Planned Maintenance System support

Source: Mean Time Between Failure (MTBF)

[22] Mean Time Between Failure (MTBF)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:mbtf

Last update: **2020/11/14 13:47**

Mean Time To Failure (MTTF)

Return to Glossary

Mean Time To Failure (MTTF) is the length of time a device or other product is expected to last in operation. MTTF is one of many ways to evaluate the reliability of pieces of hardware or other technology.

Source: Mean Time To Failure (MTTF)

Mean Time To Repair (MTTR)

Return to Glossary

Mean Time To Repair (MTTR) is a measure of the maintainability of a repairable item, which tells the average time required to repair a specific item or component and return it to working status. It is a basic measure of the maintainability of equipment and parts. This includes the notification time, diagnosis and the time spent on actual repair as well as other activities required before the equipment can be used again.

Mean time to repair is also known as Mean Repair Time.

Source: Mean Time To Repair (MTTR)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:mttr

Last update: **2020/11/14 13:34**

Media Access Control (MAC)

Return to Glossary

Media Access Control (MAC) is a sublayer of the Data Link Layer (DLL) in the seven-layer Open Systems Interconnection (OSI) Model. MAC is responsible for the transmission of data packets to and from the Network Interface Card (NIC), and to and from another remotely shared channel.

Message-Oriented Middleware (MOM)

Return to Glossary

Message-Oriented Middleware (MOM) is software or hardware infrastructure supporting sending and receiving messages between distributed systems. MOM allows application modules to be distributed over heterogeneous platforms and reduces the complexity of developing applications that span multiple operating systems and network protocols. The middleware creates a distributed communications layer that insulates the application developer from the details of the various operating systems and network interfaces. APIs that extend across diverse platforms and networks are typically provided by MOM.

This middleware layer allows software components (applications, Enterprise JavaBeans, servlets, datawriters, datareaders and other components) that have been developed independently and that run on different networked platforms to interact with one another. Applications distributed on different network nodes use the application interface to communicate. In addition, by providing an administrative interface, this new, virtual system of interconnected applications can be made reliable and secure.

Source: Message-Oriented Middleware (MOM)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:mom

Last update: **2020/11/13 15:29**

Message Queue(MQ)

Return to Glossary

Message Queue(MQ) is a software engineering component used for communication between processes or between threads within the same process. … Message queues are used within operating systems or applications as a way for programs to communicate with one another. Source: Message Queue(MQ)

Metadata

Return to Glossary

Metadata is data about data. In other words, it is data that is used to describe another item's content.

The term metadata is often used in the context of Web pages, where it describes page content for a search engine.

Source: Metadata

Microcontroller

Return to Glossary

A **Microcontroller** is a small computer on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

Source: https://en.wikipedia.org/wiki/Microcontroller

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:microcontroller

Last update: **2021/06/15 13:54**

Micropayment Channel

Return to Glossary

See **Payment Channel**

Middleware

Return to Glossary

Middleware is a software layer situated between applications and operating systems. Middleware is typically used in distributed systems where it simplifies software development by doing the following:

- Hides the intricacies of distributed applications
- Hides the heterogeneity of hardware, operating systems and protocols
- Provides uniform and high-level interfaces used to make interoperable, reusable and portable applications
- Provides a set of common services that minimizes duplication of efforts and enhances collaboration between applications

Source: Middleware

Miner Node

Return to Glossary

A **Miner Node** on creates blocks in the blockchain which the nodes keep. Basically, the miner works on transactions by coming up with the best combination (hash) to store that information. In Bitcoin, Miners spend about 10 minutes working on a problem, but nodes keep that result forever after in the database and verify it with others. Miners don't need to know about prior blocks (except for the prior one) with very few exceptions.

So, a miner is completely different than a full node. It's not comparing the same like things. Full vs Light is comparing two like things - fruit (apple and orange). Miner vs FullNode is comparing two totally different things (apple and fence).

Source: Miner Node

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:miner

Last update: **2020/11/13 09:35**

Mission Assurance (MA)

Return to Glossary

Mission Assurance (MA) is the ability of operators to achieve their mission, continue critical processes, and protect people and assets in the face of internal and external attack (both physical and cyber), unforeseen environmental or operational changes, and system malfunctions.

Source: MITRE Systems Engineering Guide

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:mission_assurance

Last update: **2020/11/13 15:29**

Mission Critical System

Return to Glossary

A **Mission Critical System** is a system that is essential to the survival of a business or organization. When a mission critical system fails or is interrupted, business operations are significantly impacted.

A mission-critical system is also known as mission essential equipment and mission critical application.

Source: Mission Critical System

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:missioncritical

Last update: **2021/06/15 13:55**

Modem

Return to Glossary

A **Modem** Modems (modulators-demodulators) are used to transmit digital signals over analog telephone lines. Thus, digital signals are converted by the modem into analog signals of different frequencies and transmitted to a modem at the receiving location. The receiving modem performs the reverse transformation and provides a digital output to a device connected to a modem, usually a computer. The digital data is usually transferred to or from the modem over a serial line through an industry standard interface, RS-232. Many telephone companies offer DSL services, and many cable operators use modems as end terminals for identification and recognition of home and personal users. Modems work on both the Physical and Data Link layers.

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:modem

Last update: **2020/11/14 16:00**

Modifiability

Return to Glossary

See 4.2.3.4 Modifiability

Modifiability is the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/57-maintainability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:modifiability

Last update: **2020/11/14 20:37**

Modified Off-The-Shelf (MOTS)

Return to Glossary

Modified Off-The-Shelf (MOTS) is a type of software solution that can be modified and customized after being purchased from the software vendor. MOTS is a software delivery concept that enables source code or programmatic customization of a standard prepackaged, market-available software.

Source: Modified Off-The-Shelf (MOTS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:mots

Last update: **2020/11/14 20:23**

Modularity

Return to Glossary

See 4.2.3.1 Modularity

Modularity is the degree to which a system's components may be separated and recombined, often with the benefit of flexibility and variety in use. The concept of modularity is used primarily to reduce complexity by breaking a system into varying degrees of interdependence and independence across and “hide the complexity of each part behind an abstraction and interface.”

Source: https://en.wikipedia.org/wiki/Modularity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:modularity

Last update: **2020/11/14 18:32**

Module

Return to Glossary

A **Module** is a separate, interchangeable, self contained component that represents a unit of functionality. It is best to define a Module what it does:

- It encapsulates code and data into a single unit (usually a file) that implements a specific functionality
- It provides an interface as a contract between the users of the module (i.e., clients) to access the functionality in an consistent way
- It allows for the functionality to be plug-able into other modules (i.e., clients) that use the interface
- It is packaged in a single unit so that it can be easily deployed. Generally it is stored as a file, but it could be an entry into a database (i.e., stored procedure), a datastream, etc.

For example, `dapper.net` encapsulates database access. It has an Application Programming Interface (API) to access its functionality. It is a single file that can be plugged into a source tree.

Source: https://softwareengineering.stackexchange.com/questions/167859/what-actually-is-a-module-in-software-engineering

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:module

Last update: **2020/11/14 14:18**

Monitoring Software

Return to Glossary

Monitoring Software observes and tracks the operations and activities of users, applications and network services on a computer or enterprise systems. This type of software provides a way to supervise the overall processes that are performed on a computing system, and provides reporting services to the system or network administrator.

Source: https://www.techopedia.com/definition/4313/monitoring-software

Motherboard

Return to Glossary

Motherboard (also called **mainboard**, main circuit board, system board, baseboard, planar board, logic board, or mobo) is the main printed circuit board (PCB) in general-purpose computers and other expandable systems. It holds and allows communication between many of the crucial electronic components of a system, such as the central processing unit (CPU) and memory, and provides connectors for other peripherals. Unlike a backplane, a motherboard usually contains significant sub-systems, such as the central processor, the chipset's input/output and memory controllers, interface connectors, and other components integrated for general use.

Motherboard means specifically a PCB with expansion capabilities. As the name suggests, this board is often referred to as the “mother” of all components attached to it, which often include peripherals, interface cards, and daughtercards: sound cards, video cards, network cards, hard drives, and other forms of persistent storage; TV tuner cards, cards providing extra USB or FireWire slots; and a variety of other custom components. Dell Precision T3600 System Motherboard , used in professional CAD Workstations. Manufactured in 2012

Similarly, the term mainboard describes a device with a single board and no additional expansions or capability, such as controlling boards in laser printers, television sets, washing machines, mobile phones, and other embedded systems with limited expansion abilities.

Source: https://en.wikipedia.org/wiki/Motherboard

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:motherboard

Last update: **2021/06/15 13:55**

Multi-core Processor

Return to Glossary

A **Multi-core Processor** is an integrated circuit (IC) to which two or more processors have been attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks (see parallel processing). A dual core set-up is somewhat comparable to having multiple, separate processors installed in the same computer, but because the two processors are actually plugged into the same socket, the connection between them is faster. [23)]

Source: https://searchdatacenter.techtarget.com/definition/multi-core-processor

[23)] Rouse, Margret; Definition of multi-core processor , WhatIs.com, Accessed 8 December 2020, https://searchdatacenter.techtarget.com/definition/multi-core-processor

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:multicore

Last update: **2021/06/15 15:37**

Multi-Signature (multisig)

Return to Glossary

Multi-Signature (multisig) isis a wallet configuration that requires at least two keys to authorize a transaction. Commonly used by cryptocurrency exchanges to ensure funds can't be moved by a rogue employee, multisig also has applications for end-users. If you're seeking to enhance the security of your noncustodial bitcoin wallet, multi-signature might be the answer.

Source: How to Use Multisig to Keep Your Coins Ultra-Safe

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:multi-signature

Last update: **2020/11/13 15:29**

Must (Requirement)

Return to Glossary

Must is not per se the specification of a requirement, in fact **Must** should not be use because no one has defined how must is different from shall. Also, shall has held up in court, must has not. Granted, must does sounds stronger, right? I mean, if your boss tells you that you “must” do something, well you are going to do it. But, when writing requirements, keep things simple and just use shall.

Source: https://reqexperts.com/2012/10/09/using-the-correct-terms-shall-will-should/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:m:must_req

Last update: **2021/06/15 13:57**

N

Return to Glossary

Create a Glossary entry starting with 'N' **Word or Expression** →	□	Add page

- N-Tier Architecture
- NATO Off-The-Shelf (NOTS)
- Near-Field-Communication (NFC)
- Network Appliance
- Network Attached Storage (NAS)
- Network Cabling
- Network Device
- Network Interface Card (NIC)
- Network Layer
- Network Node
- Network Performance
- Network Security
- Network Topology
- Network Traffic Analyzer
- Node
- Node Network
- Non-Disclosure Agreement (NDA)
- Non-Functional Requirements
- Non-Repudiation
- Normalization
- NoSQL

=-
=-

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:start

Last update: **2021/06/14 21:37**

N-Tier Architecture

Return to Glossary

N-Tier Architecture (also known as **Multi-Tier Architecture**) is a client-server architecture concept in software engineering where the presentation, processing and data management functions are both logically and physically separated. These functions are each running on a separate machine or separate clusters so that each is able to provide the services at top capacity since there is no resource sharing. This separation makes managing each separately easier since doing work on one does not affect the others, isolating any problems that might occur.

Source: https://www.techopedia.com/definition/17185/n-tier-architecture

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:n-tier

Last update: **2020/11/14 20:40**

NATO Off-The-Shelf (NOTS)

Return to Glossary

**NATO Off-The-Shelf (NOTS) or niche off-the-shelf, depending on the context) product is developed by NC3A (for NATO Consultation, Command, and Control) to meet specific requirements for NATO. In the more general context, niche off-the-shelf refers to vendor-developed software that is for a specialized and narrow market segment, in comparison to the broad market for Commercial Off-The-Shelf (COTS) products.

Source: NATO Off-The-Shelf (NOTS)

Near-Field-Communication (NFC)

Return to Glossary

Near-Field-Communication (NFC) is a set of communication protocols for communication between two electronic devices over a distance of 4 cm (11⁄2 in) or less.[1] NFC offers a low-speed connection with simple setup that can be used to bootstrap more-capable wireless connections.

NFC devices can act as electronic identity documents and keycards. They are used in contactless payment systems and allow mobile payment replacing or supplementing systems such as credit cards and electronic ticket smart cards. This is sometimes called NFC/CTLS or CTLS NFC, with contactless abbreviated CTLS. NFC can be used for sharing small files such as contacts, and bootstrapping fast connections to share larger media such as photos, videos, and other files.

Source: https://en.wikipedia.org/wiki/Near-field_communication

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:nfc

Last update: **2021/06/15 13:58**

Network Appliance

Return to Glossary

Network Appliance is a type of computing appliance that aids in the flow of information to other network-connected computing devices. Services that may be provided by a network appliance include firewall functions, caching, authentication, network address translation and IP address management.

Source: https://www.gartner.com/en/information-technology/glossary/network-appliance

Network Attached Storage (NAS)

Return to Glossary

Network Attached Storage (NAS) is a Network Appliance that is a dedicated server, also referred to as an appliance, used for file storage and sharing. NAS is a hard drive attached to a network, used for storage and accessed through an assigned network address. It acts as a server for file sharing but does not allow other services (like emails or authentication). It allows the addition of more storage space to available networks even when the system is shutdown during maintenance.

NAS is a complete system designed for heavy network systems, which may be processing millions of transactions per minute. NAS provides a widely supported storage system for any organization requiring a reliable network system.

Source: https://www.techopedia.com/definition/26197/network-attached-storage-nas

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:nas

Last update: **2021/06/15 13:59**

Network Cabling

Return to Glossary

Network Cabling are used to connect and transfer data and information between computers, routers, switches and storage area networks . These cables are essentially the carrier or media through which data flows.

There are different types of communications cables, and the appropriate type to use will depend on the structure and topology of the overall architecture of the system. The most commonly used types of communications cables are dominated by what is referred to as “twisted pair cable”. In local area networks; typically office environments, retail and commmercial sites, copper commincations cabling, i.e.,twisted pair cable is by far the most commonly used type of cable.

Some common network cabling standards are:

- Category 5 (Cat-5)
- Category 6 (Cat-6)
- Category 7 (Cat-7)
- Category 8 (Cat-8)

Source: https://totalcommstraining.com/what-is-network-cabling/

Network Device

Return to Glossary

Network Device is one of the following devices:

- Hub
- Switch
- Router
- Bridge
- Gateway
- Modem
- Repeater
- Network Appliance

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:netdev

Last update: **2020/12/09 11:05**

Network Interface Card (NIC)

Return to Glossary

A Network Interface Card (NIC) is a hardware component without which a computer cannot be connected over a network. It is a circuit board installed in a computer that provides a dedicated network connection to the computer. It is also called network interface controller, network adapter or LAN adapter.

The purpose of the NIC:

- Allows both wired and wireless communications.
- Allows communications between computers connected via Local Area Network (LAN) as well as communications over large-scale Wide Area Network (WAN) networks through Internet Protocol (IP).
- Supports both a physical layer and a data link layer device, i.e. it provides the necessary hardware circuitry so that the physical layer processes and some data link layer processes can run on it.

Source: https://www.tutorialspoint.com/what-is-network-interface-card-nic

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:nic

Last update: **2021/06/15 14:01**

Network Layer

Return to Glossary

The **Network Layer** is the third level of the Open Systems Interconnection (OSI) Model and the layer that provides data routing paths for network communication. Data is transferred in the form of packets via logical network paths in an ordered format controlled by the network layer.

Logical connection setup, data forwarding, routing and delivery error reporting are the Network Layer's primary responsibilities.

Source: Network Layer

Network Node

Return to Glossary

A **Network Node** is a connection point that can receive, create, store or send data along distributed network routes. Each network node – whether it's an endpoint for data transmissions or a redistribution point – has either a programmed or engineered capability to recognize, process and forward transmissions to other network nodes.

Source: Network Node

Network Performance

Return to Glossary

Network Performance is the analysis and review of collective network statistics, to define the quality of services offered by the underlying computer network.

It is a qualitative and quantitative process that measures and defines the performance level of a given network. It guides a network administrator in the review, measure and improvement of network services.

Source: Network Performance

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:netperform

Last update: **2020/11/16 21:06**

Network Security

Return to Glossary

Network Security is an over-arching term that describes that the policies and procedures implemented by a network administrator to avoid and keep track of unauthorized access, exploitation, modification, or denial of the network and network resources.

This means that a well-implemented network security blocks viruses, malware, hackers, etc. from accessing or altering secure information.

Source: https://www.techopedia.com/definition/24783/network-security

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:networksecurity

Last update: **2020/11/14 14:31**

Network Topology

Return to Glossary

Network Topology refers to the physical or logical layout of a network. It defines the way different nodes are placed and interconnected with each other. Alternately, network topology may describe how the data is transferred between these nodes.

There are two types of network topologies: physical and logical. Physical topology emphasizes the physical layout of the connected devices and nodes, while the logical topology focuses on the pattern of data transfer between network nodes.

Source: Network Topology

Network Traffic Analyzer

Return to Glossary

Network_traffic_analyzer (NTA) intercepts, records and analyzes network traffic communications. The communications are scanned for patterns that indicate security threats and provides methods to respond to security threats.

Source: Network Traffic Analyzer

Node

Return to Glossary

A **Node** is an individual computer that participates as an equal within the Node Network. A node is sometimes referred to as a **peer** within a Peer to Peer (P2P) network. Nodes receive input, perform one or more operations on those inputs; and returns an output.[24]

Source: Node

[24] Not all DIDOs use "mining" as originally defined in Bitcoin as Proof of Work (PoW) and they may not need a full set blockchain transaction to verify transactions.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:node

Last update: **2021/04/16 17:04**

Node Network

Return to Glossary

A **Node Network** is a collection of computers or Modes that are interconnected and communicating over a network of equally privileged computers (i.e., there are no central authoritative computers). The **Node network** is sometimes referred to as a **peer-to-peer (P2P) network**. [25] [26] [27] [28]

[25] “What is a Bitcoin node?”, Nate Eldredge, 14, December 2013, https://bitcoin.stackexchange.com/questions/18736/what-is-a-bitcoin-node

[26] “What are Ethereum Nodes And Sharding?”, Ameer Rosic, 2017, https://blockgeeks.com/guides/what-are-ethereum-nodes-and-sharding/

[27] “IOTA Full Node—That’s Why a Full Node Is Important for IOTA!”, Marko Vidrih, February 2019, https://medium.com/altcoin-magazine/iota-full-node-thats-why-a-full-node-is-important-for-iota-1c46280d7712

[28] “Introduction to IPFS: Run Nodes on Your Network, with HTTP Gateways”, Ross Bulat, 3 December 2018, https://medium.com/@rossbulat/introduction-to-ipfs-set-up-nodes-on-your-network-with-http-gateways-10e21ea689a4

Non-Disclosure Agreement (NDA)

Return to Glossary

A **Non-Disclosure Agreement (NDA)** is a legally binding contract that establishes a confidential relationship. The party or parties signing the agreement agree that sensitive information they may obtain will not be made available to any others.

Non-disclosure agreements are common for businesses entering into negotiations with other businesses. They allow the parties to share sensitive information without fear that it will end up in the hands of competitors. In this case, it may be called a mutual non-disclosure agreement.

Source: https://www.investopedia.com/terms/n/nda.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:nda

Last update: **2020/11/15 19:54**

Non-Functional Requirements

Return to Glossary

Non-Functional Requirements specify how the system should do it. Non-functional requirements do not affect the basic functionality of the system (hence the name, non-functional requirements). Even if the non-functional requirements are not met, the system will still perform its basic purpose.

If a system will still perform without meeting the non-functional requirements, why are they important? The answer is usability. Non-functional requirements define system behavior, features, and general characteristics that affect the user experience. How well non-functional requirements are defined and executed determines how easy the system is to use, and is used to judge system performance. Non-functional requirements are product properties and focus on user expectations.

See also Functional Requirements.

Source: Non-Functional Requirements

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:nonfuncreq

Last update: **2020/11/13 15:29**

Non-Repudiation

Return to Glossary

See 2.2.4.3 Non-Repudiation

Non-Repudiation means to ensure that a transferred message has been sent and received by the parties claiming to have sent and received the message. Nonrepudiation is a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

Source: https://www.webopedia.com/TERM/N/nonrepudiation.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:nonrepudiation

Last update: **2020/11/16 12:13**

Normalization

Return to Glossary

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and update anomalies using Normal forms to eliminate or reduce redundancy.

Source: https://www.geeksforgeeks.org/normal-forms-in-dbms/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:normalization

Last update: **2020/11/14 18:29**

NoSQL

Return to Glossary

NoSQL refers to a Datastore that provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but the name “NoSQL” was only coined in the early 21st century, triggered by the needs of Web 2.0 companies.

Source: https://en.wikipedia.org/wiki/NoSQL

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:n:nosql

Last update: **2021/06/15 14:03**

O

Return to Glossary

Create a Glossary entry starting with 'O' **Word or Expression** →		Add page

- Object
- Object Management Group® (OMG)
- Object-Oriented Programming (OOP)
- Objective
- One-Time PIN (OTP)
- Ontology
- Open Database Connectivity (ODBC)
- Open Source Software (OSS)
- Open Systems Interconnection (OSI) Model
- Operability
- Operating System (OS)
- Operational transformation (OT)
- Operator
- Oracle
- Owner
- Ownership

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:start

Last update: **2021/06/14 21:37**

Object

Return to Glossary

An **Object** is a class of things one thinks about first in designing a program and they are also the units of code that are eventually derived from the process. In Object-Oriented Programming (OOP), each thing is made into a generic Class and instances of the Class are referred to as an object. Classes are specific or generic in nature. Specific Classes are specifically designed to represent a specific class of things. Generic Classes are templates of how an object can be manipulated. (See: https://searchapparchitecture.techtarget.com/definition/object).

For example, there is a thing in the system defined specifically to caption the concepts of a car (i.e., wheels, chassis, steering wheel, etc.). There may also be a Generic Class that represents a collection of things. The Generic Class can be applied to many different classes to create a collection of things (i.e., cars).

Source: Reference Architecture (RA)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:object

Last update: **2021/06/15 14:04**

Object-Oriented Programming (OOP)

Return to Glossary

Object-Oriented Programming (OOP) refers to a type of computer programming (software design) in which programmers define the data type of a data structure, and also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

Source: https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:oop

Last update: **2020/11/14 20:09**

Objective

Return to Glossary

An **Objective** is a lower level statements that describe the specific, tangible products and deliverables that the project will deliver.

Source: Objective

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:objective

Last update: **2020/12/20 20:57**

Object Management Group® (OMG)

Return to Glossary

The **Object Management Group® (OMG)** is an international, open membership, not-for-profit technology standards consortium.

Founded in 1989, OMG standards are driven by vendors, end-users, academic institutions and government agencies. OMG Task Forces develop enterprise integration standards for a wide range of technologies and an even wider range of industries.

Source: Object Management Group® (OMG)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:omg

Last update: **2021/06/15 14:04**

One-Time PIN (OTP)

Return to Glossary

A **One-Time PIN (OTP)** is a code that is valid for only one login session or transaction using a mobile phone. There are a number of ways to deliver one-time passwords and pins with the two most common and secure ways being through proprietary tokens and mobile phones. …

Source: https://www.infobip.com/glossary/otp-one-time-pin-code

Ontology

Return to Glossary

An **Ontology** may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms. It is the specification of conceptualisations, used to help programs and humans share knowledge.

Source: http://www.cs.man.ac.uk/~stevensr/onto/node3.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:ontology

Last update: **2021/06/15 14:05**

Open Database Connectivity (ODBC)

Return to Glossary

Open Database Connectivity (ODBC) is an Application Programming Interface (API) a Microsoft de facto standard Open Database Connectivity (ODBC) for accessing DataBase Management System (DBMS). The designers of ODBC aimed to make it independent of database systems and Operating System (OS). An application written using ODBC can be ported to other platforms, both on the client and server side, with few changes to the data access code.

ODBC accomplishes DBMS independence by using an ODBC driver as a translation layer between the application and the DBMS. The application uses ODBC functions through an ODBC driver manager with which it is linked, and the driver passes the query to the DBMS. An ODBC driver can be thought of as analogous to a printer driver or other driver, providing a standard set of functions for the application to use, and implementing DBMS-specific functionality. An application that can use ODBC is referred to as “ODBC-compliant”. Any ODBC-compliant application can access any DBMS for which a driver is installed. Drivers exist for all major DBMSs, many other data sources like address book systems and Microsoft Excel, and even for text or Comma Separated Values (CSV) files.

ODBC was originally developed by Microsoft and Simba Technologies during the early 1990s, and became the basis for the Call Level Interface (CLI) or Command Line Interface (CLI) standardized by SQL Access Group in the Unix and mainframe field. ODBC retained several features that were removed as part of the CLI effort. Full ODBC was later ported back to those platforms, and became a de facto standard considerably better known than CLI. The CLI remains similar to ODBC, and applications can be ported from one platform to the other with few changes.

Source: https://en.wikipedia.org/wiki/Open_Database_Connectivity

Open Source Software (OSS)

Return to Glossary

Open source software (OSS) doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria: free distribution; source code; derived works; integrity of the author's source code; no discrimination against persons or groups; no discrimination against fields of endeavor; distribution license; license must not be specific to a product; license must not restrict other software; and license must be technology-neutral.

Source: The Open Source Definition, 22 March 2007

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:oss

Last update: **2020/11/13 03:00**

Open Systems Interconnection (OSI) Model

Return to Glossary

The **Open Systems Interconnection (OSI) Model** is a conceptual and logical layout that defines network communication used by systems open to interconnection and communication with other systems.

The model is broken into seven subcomponents, or layers, each of which represents a conceptual collection of services provided to the layers above and below it. The OSI Model also defines a logical network and effectively describes computer packet transfer by using different layer protocols.

The OSI Model may also be referred to as the seven-layer OSI Model or the seven-layer model.

The Seven Layers are:

Layer	Name
7	Application Layer
6	Presentation Layer
5	Data Link Layer (DLL)
4	Network Layer
3	Session Layer
2	Physical Layer
1	Transport Layer

Source: Open Systems Interconnection (OSI) Model

Operability

Return to Glossary

Operability is degree to which a product or system has attributes that make it easy to operate and control.User error protection. Degree to which a system protects users against making errors.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

Operating System (OS)

Return to Glossary

An **operating system (OS)**, in its most general sense, is software that allows a user to run other applications on a computing device. While it is possible for a software application to interface directly with hardware, the vast majority of applications are written for an OS, which allows them to take advantage of common libraries and not worry about specific hardware details.

Source: Operating System (OS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:os

Last update: **2020/11/13 03:00**

Operational transformation (OT)

Return to Glossary

Operational transformation (OT) is a technology for supporting a range of collaboration functionalities in advanced collaborative software systems. OT was originally invented for consistency maintenance and concurrency control in collaborative editing of plain text documents. Its capabilities have been extended and its applications expanded to include group undo, locking, conflict resolution, operation notification and compression, group-awareness, HTML/XML and tree-structured document editing, collaborative office productivity tools, application-sharing, and collaborative computer-aided media design tools.[1] In 2009 OT was adopted as a core technique behind the collaboration features in Apache Wave and Google Docs.

Source: Operational transformation (OT)

Operator

Return to Glossary

An **Operator** is an individual, entity, or joint operation who is determined by County Committees (COC) as being in general control of the farming operations on the farm for the current year. Source: USDA - Farm Records and Reconstitutions page 21

Oracle

Return to Glossary

An **Oracle**, in the context of blockchains and smart contracts, is an agent that finds and verifies real-world occurrences and submits this information to a blockchain to be used by smart contracts.

Source: Blockchain Oracles

Owner

Return to Glossary

An **Owner** is an individual or entity who has legal ownership of farmland, including individuals or entities that are any of the following:

- buying farmland under a contract for deed
  Note: USDA Office of the General Counsel (OGC), Regional Attorney will review contracts that are questionable before changing USDA Farm Service Agency (FSA) ownership records.
- retaining a life estate in the property
- purchasing a farm in a foreclosure proceeding and both of the following apply:
- the redemption period has not passed
- the original owner has not redeemed the property
- a spouse in a community property State
- spouses owning property jointly.

County Committees (COC) will require specific proof of ownership when land ownership is transferred.

Source: USDA - Farm Records and Reconstitutions, page 22

Ownership

Return to Glossary

Ownership is …

Source: URI

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:o:ownership

Last update: **2021/06/15 14:09**

P

Return to Glossary

Create a Glossary entry starting with 'P' **Word or Expression** →	Add page

- Package Manager
- Packet Loss
- Packet Switched Network (PSN)
- Parallel Processing
- Parliamentary Authority
- Partition
- Payment Channel
- Pedigree
- Peer to Peer (P2P)
- Performance
- Performance Efficiency Measure
- Performance or Functional Specifications
- Permissioned Networks
- Permissionless Networks
- Permissive Open Source Software
- Personal Identifiable Information (PII)
- Physical Integrity
- Physical Layer
- Physical Security
- Planning Level
- Platform
- Platform Independent Model (PIM)
- Platform Security
- Platform Specific Model (PSM)
- Platform-as-a-Service (PaaS)
- Plug In
- Point-to-Point
- Policies and Procedures (P&P)
- Policy
- Port Number
- Portability
- Portable Operating System Interface (POSIX)
- Presentation Layer
- Principle
- Principles
- Private Network
- Privileges
- Procedural Language

- Procedure
- Processor
- Programming Language
- Project Management Software
- Proof of Authority (PoA)
- Proof of Stake (PoS)
- Proof of Work (PoW)
- Protocol
- Protocol Layer
- Provenance
- Public Key Infrastructure (PKI)
- Public Network
- Publish-Subscribe
- Publisher

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:start

Last update: **2021/06/14 21:38**

Package Manager

Return to Glossary

A **Package Manager** or package-management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner.

A package manager deals with packages, distributions of software and data in archive files. Packages contain Metadata, such as the software's name, description of its purpose, version number, vendor, checksum (preferably a cryptographic hash function), and a list of dependencies necessary for the software to run properly. Upon installation, metadata is stored in a local package database. Package managers typically maintain a database of software dependencies and version information to prevent software mismatches and missing prerequisites. They work closely with software repositories, binary repository managers, and app stores.

Package managers are designed to eliminate the need for manual installs and updates. This can be particularly useful for large enterprises whose operating systems are typically consisting of hundreds or even tens of thousands of distinct software packages.

Source: https://en.wikipedia.org/wiki/Package_manager

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:packagemanager

Last update: **2020/11/13 03:01**

Packet Loss

Return to Glossary

Packet Loss is used when describing a Packet Switched Network (PSN) and refers to the amount of data (number of packets) that fails to arrive at its intended destination. Network administrators consider this metric when looking at the efficacy and performance of data systems.

Source: Packet Loss

Packet Switched Network (PSN)

Return to Glossary

A **Packet Switched Network (PSN)** is a type of computer communications network that groups and sends data in the form of small packets. It enables the sending of data or network packets between a source and destination node over a network channel that is shared between multiple users and/or applications.

A packet switched is also known as a connectionless network, as it does not create a permanent connection between a source and destination node.

Source: Packet Switched Network (PSN)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:psn

Last update: **2021/06/15 14:09**

Parallel Processing

Return to Glossary

Parallel Processing is a method in computing of running two or more processors (CPUs) to handle separate parts of an overall task. Breaking up different parts of a task among multiple processors will help reduce the amount of time to run a program. Any system that has more than one CPU can perform parallel processing, as well as multi-core processors which are commonly found on computers today.

Parallel processing is commonly used to perform complex tasks and computations. Data scientists will commonly make use of parallel processing for compute and data-intensive tasks.[29]

Source: https://searchdatacenter.techtarget.com/definition/parallel-processing

[29] Rouse, Margret; Definition of parallel processing, WhatIs.com, Accessed 8 December 2020, https://searchdatacenter.techtarget.com/definition/parallel-processing

Parliamentary Authority

Return to Glossary

Parliamentary Authority is the rulebook used to conduct business within a group. Robert’s Rules of Order is not the only one, but it is a common option used to help facilitate the smooth functioning of an assembly and provide a firm basis for resolving questions of procedure.

Source: Parliamentary Authority

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:parliamentary_authority

Last update: **2020/11/13 03:01**

Partition

Return to Glossary

See QoS Policy

A sequence of logical “namespaces” for topics. The default setting is an empty sequence, indicating the default Partition

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:partition

Last update: **2021/06/15 14:10**

Payment Channel

Return to Glossary

Payment Channel is a medium between two or more parties that allows them to transact bitcoins amongst them a number of times without sending all these transactions to the base layer, i.e., Bitcoin's blockchain.

A payment channel essentially is a 2-of-2 Multi-Signature (multisig) wallet where parties interested in transacting with each other commit funds and this is called funding transaction that happens on-chain.

Once this on-chain transaction is complete, many transactions can happen between the two parties involved in the payment channel. But the transactions can only happen when both the parties sign thus updating the state of the channel (or you can say an offline shared ledger between them)

Thus many signed but broadcasted transactions can be exchanged between the parties involved in the payment channel.

In simple terms, you can think of a payment channel as duct or pipe which is open from both the ends and there are two individuals on each side with 10 pearls with them.

So through this duct or pipe, these guys can send the pearls to each other number of times and when they don't want to transact they can simply close the duct or pipe.

Source: Payment Channel

Pedigree

Return to Glossary

Pedigree is meta-data about a record of the ancestry of data and may include metric estimates about the reliability and confidence in the data. In other words, pedigree is about the “who” and “when” of ownership, transfer and transformation of data.

Source: OMG

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:pedigree

Last update: **2020/11/13 13:13**

Peer to Peer (P2P)

Return to Glossary

Peer to Peer (P2P) is a network where the “peers” are computer systems which are connected to each other via the Internet. Files can be shared directly between systems on the network without the need of a central server. In other words, each computer on a P2P network becomes a file server as well as a client.

Source: Peer to Peer (P2P)

Performance

Return to Glossary

See 2.2.8 Performance

Performance is the ability of a system to accomplish the required functionality at or under the required specification limits. The limits are generally provided relative to time. For example, so-many transactions per second, so-many updates per millisecond, so-many recorded entries per second, etc. The specifications can also include accuracy, precision or even efficiency of other dependent systems as requirements.

Source: Performance

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:performance

Last update: **2021/05/13 17:10**

Performance Efficiency Measure

Return to Glossary

Performance Efficiency Measure assesses characteristics that affect an application's response behavior and use of resources under stated conditions (ISO/IEC 25010). Performance Efficiency affects customer satisfaction, workforce productivity, application scalability, response-time degradation, and inefficient use of processing or storage resources. The Performance Efficiency of an application lies in each individual component‘s performance, as well as in the effect of each component on the behavior of the chain of components comprising a transaction in which it participates.

Source: OASIS A Brief Introduction to XACML - 14 March 2003

Performance or Functional Specifications

Return to Glossary

Performance Specifications is a document specifying operational requirements of a component or installation. In other words, a performance specification informs the contractor what the final installed product capabilities are. In contrast, see Conformance Specification.

In this case the buyer simply explains problems the product should solve and the supplier provides such. In essence this is less rigid than conformance specification.

Examples of what typical specifications may be like include;

- Required quality levels
- Required safety levels and controls

Compared to conformance specification, performance specification has a number of ADVANTAGES such as;

- The specifications are easier to draft
- The efficacy of specifications does not depend on technical knowledge of the buyer
- Supplier can use their creativity to develop the products
- Greater share of specification risk is borne by the supplier.

When is it advisable to use performance specification?

- When the supplier has more technical or relevant skills than that of the buyer
- When technology is constantly changing in the suppliers industries in which case it will be hard to specify methodologies
- When there is a clear criterion for evaluating alternative solutions suggested by the suppliers competing for the contract

When there is enough time to assess the functionality of the product as proposed.

Source: http://zeritenetwork.com/typesofspecificationswhencontracting/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:performancespec

Last update: **2021/03/05 16:21**

Permissioned Networks

Return to Glossary

Permissioned blockchains allow a mixed bag between the Public Network and Private Network with lots of customization options. The available options include allowing anyone to join the permissioned network after suitable verification of their identity, and allocation of select and designated permissions to perform only certain activities on the network. [30)]

[30)] “Public, Private, Permissioned Blockchains Compared”, Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

Permissionless Networks

Return to Glossary

Permissionless Networks require no permission to use it. In other words, there is no barrier to entry to use it. Anyone can run a node, run mining software/hardware, access a wallet and write data onto and transact within the blockchain (as long as they follow the rules of the blockchain). There is no way to censor anyone, ever, on the permissionless bitcoin blockchain.[31]

[31] “Permissioned vs. Permissionless blockchains: Who will win and will it matter?”, Dustin D, 22 March 2018, Permissionless

Permissive Open Source Software

Return to Glossary

Permissive Open Source Software is simply a non-copyleft open source license — one that guarantees the freedoms to use, modify, and redistribute, but that permits proprietary derivative works. See the copyleft entry for more information.

Source: What is a "permissive" Open Source license?

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:permissive_oss

Last update: **2020/11/13 13:13**

Personal Identifiable Information (PII)

Return to Glossary

Personal Identifiable Information (PII) as defined in OMB Memorandum M-07-1616 refers to information that can be used to distinguish or trace an individual’s identity, either alone or when combined with other personal or identifying information that is linked or linkable to a specific individual. The definition of PII is not anchored to any single category of information or technology. Rather, it requires a case-by-case assessment of the specific risk that an individual can be identified. In performing this assessment, it is important for an agency to recognize that non-PII can become PII whenever additional information is made publicly available - in any medium and from any source - that, when combined with other available information, could be used to identify an individual.

Source: https://www.gsa.gov/reference/gsa-privacy-program/rules-and-policies-protecting-pii-privacy-act

Physical Integrity

Return to Glossary

Physical Integrity is the protection of data’s wholeness and accuracy as it’s stored and retrieved. When natural disasters strike, power goes out, or hackers disrupt database functions, physical integrity is compromised. Human error, storage erosion, and a host of other issues can also make it impossible for data processing managers, system programmers, applications programmers, and internal auditors to obtain accurate data.

Source: Physical Integrity

Physical Layer

Return to Glossary

The **Physical Layer** is the first layer of the Open Systems Interconnection (OSI) Model. The Physical Layer deals with bit-level transmission between different devices and supports electrical or mechanical interfaces connecting to the physical medium for synchronized communication.

This layer plays with most of the network's physical connections - wireless transmission, cabling, cabling standards and types, connectors and types, network interface cards, and more - as per network requirements. However, the physical layer does not deal with the actual physical medium (like copper, fiber).

Source: Physical Layer

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:physicallayer

Last update: **2020/11/13 13:12**

Physical Security

Return to Glossary

Physical Security describes measures designed to ensure the physical protection of Information Technology (IT) assets like facilities, equipment, personnel, resources and other properties from damage and unauthorized physical access. Physical security measures are taken in order to protect these assets from physical threats including theft, vandalism, fire and natural disasters.

Source: https://www.techopedia.com/definition/14514/physical-security

Planning Level

Return to Glossary

The **Planning Level** is the forth level of the Automation Pyramid. It uses Manufacturing Execution System (MES) to monitor the entire manufacturing process. For example, in a factory to plan for everything from raw materials to the finished products. This allows management to visualize the current state of operations and aids them in making decisions and adjust raw material orders or shipment plans based on real data received from Supervisory, Control and Field Levels.

Source: Planning Level

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:planlevel

Last update: **2020/11/15 20:22**

Platform

Return to Glossary

A **Platform** is a group of technologies that are used as a base upon which other applications, processes or technologies are developed.

In personal computing, a platform is the basic hardware (computer) and software (operating system) on which software applications can be run. This environment constitutes the basic foundation upon which any application or software is supported and/or developed.

Computers use specific central processing units (CPUs) that are designed to run specific machine language code. In order for the computer to run software applications, the applications must be in that CPU's binary-coded machine language.

Thus, historically, application programs written for one platform would not work on a different platform.

Source: https://www.techopedia.com/definition/3411/platform-computing

Platform-as-a-Service (PaaS)

Return to Glossary

Platform-as-a-Service (PaaS) is a form of cloud computing where the hardware and software platform is provided by a third party.

Source: Platform-as-a-Service (PaaS)

Platform Independent Model (PIM)

Return to Glossary

Platform Independent Model (PIM) is a model of a subsystem that contains no information specific to the platform or the technology that is used to realize it. The quality that the model is independent of the features of a platform of any particular type. Like most qualities, platform independence is a matter of degree. A view of a system from the platform independent viewpoint. A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms of similar type. [MDA Guide]. Platform Independent Viewpoint: Focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent view shows that part of the complete specification that does not change from one platform to another

Source: Platform-Independent Model (PIM)

Platform Security

Return to Glossary

Platform Security refers to the security architecture, tools and processes that ensure the security of an entire Computing Platform.

It uses bundled/unified security software, systems and processes to enable the security of a computing platform's hardware, software, network, storage and other components.

Source: https://www.techopedia.com/definition/4053/platform-security

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:platformsecurity

Last update: **2020/11/15 17:45**

Platform Specific Model (PSM)

Return to Glossary

Platform Specific Model (PSM) A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform. A view of a system from the platform specific viewpoint, combining the specifications in the PIM with the details that specify how that system uses a particular type of platform

Source: Platform-Specific Model (PSM)

Plug In

Return to Glossary

Plug-In, also called **Add-On** or extension is computer software that adds new functions to a host program without altering the host program itself. Widely used in digital audio, video, and Web browsing, plug-ins enable programmers to update a host program while keeping the user within the program's environment.

Source: https://www.britannica.com/technology/plug-in

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:plug_in

Last update: **2020/11/14 18:28**

Point-to-Point

Return to Glossary

Point-to-point is the simplest model of communication, as illustrated in Figure 1; it is a model of one-to-one communications. The telephone is an example of an everyday point-to-point communications device. To use a telephone, you must know the address (phone number) of the other party. Once a connection is established, you can have a reasonably high-bandwidth conversation. However, the telephone does not work as well if you have to talk to many people at the same time. The telephone is essentially one-to-one communication.

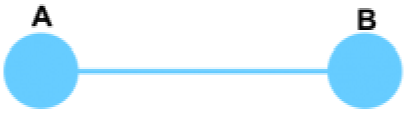


Figure 1: Point-to-Point Communication Model

Transmission Control Protocol (TCP) is a point-to-point network protocol designed in the 1970s. While it provides reliable, high bandwidth communication, TCP is cumbersome for systems with many communicating nodes.

Source: https://community.rti.com/glossary/point-point

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:point-to-point

Last update: **2021/06/15 14:13**

Policies and Procedures (P&P)

Return to Glossary

Policies and Procedures (P&P) in a company serve to define how employees are expected to behave and to detail the responsibilities of both management and employees. Company policies and procedures help to ensure that employees receive their legal and ethical entitlements. At the same time, they guarantee that an organization pays proper attention to business concerns. Source: https://www.lanast.com/policies-and-procedures-that-all-companies-should-have/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:p_p

Last update: **2021/06/15 14:13**

Policy

Return to Glossary

Policy is a precise statement which contains the set of principles acting as guidelines for achieving the goals of an organization.

Source: Policy

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:policy

Last update: **2020/11/13 13:16**

Portability

Return to Glossary

See 2.2.1 Portability

Portability, in relation to software, is a measure of how easily an application can be transferred from one computer environment to another. A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits. The meaning of the abstract term 'reasonable' depends upon the nature of the application and is often difficult to express in quantifiable units.

The phrase “to port” means to modify software and make it adaptable to work on a different computer system. For example, to port an application to Linux means to modify the program so that it can be run in a Linux environment.

Portability refers to the ability of an application to move across environments, not just across platforms. To clarify, a computer platform generally refers to the operating system and computer hardware only. A computer environment is much broader and may include the hardware, the operating system and the interfaces with other software, users and programmers.

Source: https://www.techopedia.com/definition/8921/portability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:portability

Last update: **2020/11/13 13:13**

Portable Operating System Interface (POSIX)

Return to Glossary

Portable Operating System Interface (POSIX) is defined by POSIX.1-2017 as a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level. It is intended to be used by both application developers and system implementors.

Source: IEEE 1003.1-2017 - IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications

Port Number

Return to Glossary

A **Port Number** is the logical address of each application or process that uses a network or the Internet to communicate. A port number uniquely identifies a network-based application on a computer. Each application/program is allocated a 16-bit integer port number. This number is assigned automatically by the OS, manually by the user or is set as a default for some popular applications.

Source: Port Number

Presentation Layer

Return to Glossary

The **Presentation Layer** is layer 6 of the 7-layer Open Systems Interconnection (OSI) Model. It is used to present data to the Application Layer (layer 7) in an accurate, well-defined and standardized format.

The Presentation Layer is sometimes called the Syntax Layer.

Source: Presentation Layer

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:presentlayer

Last update: **2020/11/13 13:18**

Principle

Return to Glossary

A **Principle** is an elementary assumption, concept, doctrine, maxim, or proposition generally held to be fundamental or true for a body of knowledge, conduct, procedure, or system of reasoning, and used as a basis for prediction and action. See also principles.

Source: Principles

Principles

Return to Glossary

Principles are fundamental norms, rules, or values that represent what is desirable and positive for a person, group, organization, or community, and help it in determining the rightfulness or wrongfulness of its actions. Principles are more basic than policy and objectives, and are meant to govern both. See also Principle.

Source: Principles

Private Network

Return to Glossary

A **Private Network** allows only selected entry of verified participants, like those for a private business, one can opt for a private network implementation. A participant can join such a private network only through an authentic and verified invitation, and a validation is necessary either by the network operator(s) or by a clearly defined set protocol implemented by the network.[32]

[32] “Public, Private, Permissioned Blockchains Compared”, Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:private_network

Last update: **2020/11/13 13:17**

Privileges

Return to Glossary

Privileges in the context of computer security, are the concept of only allowing users to do certain things. For example, an ordinary user is typically prevented from changing operating system files, while a system administrator is typically permitted to do so, because this is part of maintaining a computer system. Maintaining user privilege is typically accomplished through the use of administrative accounts, file permissions and Access Control List (ACL).

Source: https://www.techopedia.com/definition/16044/privilege-security

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:privileges

Last update: **2020/11/13 13:19**

Procedural Language

Return to Glossary

A **Procedural Language** is a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program. It contains a systematic order of statements, functions and commands to complete a computational task or program.

Procedural language is also known as **Imperative Language**. Source: https://www.techopedia.com/definition/8982/procedural-language

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:procedurallanguage

Last update: **2021/06/15 14:16**

Procedure

Return to Glossary

Procedure is a fixed, step-by-step sequence of activities or course of action (with definite start and end points) that must be followed in the same order to correctly perform a task. Repetitive procedures are called routines.

Source: Procedure

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:procedure

Last update: **2020/11/13 13:13**

Processor

Return to Glossary

A *Processor** is an integrated electronic circuit that performs the calculations that run a computer. A processor performs arithmetical, logical, input/output (I/O) and other basic instructions that are passed from an Operating System (OS). Most other processes are dependent on the operations of a processor.

The terms processor, Central Processing Unit (CPU) and microprocessor are commonly linked.

Source: Processor

Programming Language

Return to Glossary

Programming Language is a computer language engineered to create a standard form of commands. These commands can be interpreted into a code understood by a machine. Programs are created through programming languages to control the behavior and output of a machine through accurate algorithms, similar to the human communication process. Examples of programming languages are assembler, C/C++, C#, Java, JavaScrpipt, Python, Erlang, HTML, etc.

A programming language is also known as a programming system, computer language or computer system.

Source: Programming Language

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:programlang

Last update: **2020/11/13 13:12**

Project Management Software

Return to Glossary

Project Management Software is software used for project planning, scheduling, resource allocation and change management. It allows project managers (PMs), stakeholders and users to control costs and manage budgeting, quality management and documentation and also may be used as an administration system. Project management software is also used for collaboration and communication between project stakeholders.

Source: Project Management Software

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:projmansw

Last update: **2021/06/15 14:17**

Proof of Authority (PoA)

Return to Glossary

Proof of Authority (PoA) is a modified form of Proof of Stake (PoS) where instead of stake with the monetary value, a validator’s identity performs the role of stake. In this context, identity means the correspondence between a validator’s personal identification on the platform with officially issued documentation for the same person, i.e. certainty that a validator is exactly who that person represents to be.

Source: Proof of Authority (PoA)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:proof_of_authority_poa

Last update: **2020/11/13 13:12**

Proof of Stake (PoS)

Return to Glossary

Proof of Stake (PoS) is a way of achieving consensus by distributing validation of a block of transactions based on the number of tokens held rather than by rewarding miners like Proof of Work (PoW). Consequently, PoS is a way of validating a block that requires far less energy that Proof of Work (PoW). An important key aspect of PoS is including a degree of chance to the selection process to avoid a scenario where the richest users are always selected to validate transactions and consistently reap the rewards or to bias validation in their favour.

Source: What is Proof of Stake? (PoS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:proof_of_stake_pos

Last update: **2020/11/13 13:17**

Proof of Work (PoW)

Return to Glossary

Proof of work (PoW) describes a system that requires a not-insignificant but feasible amount of effort in order to deter frivolous or malicious uses of computing power, such as sending spam emails or launching denial of service attacks. The concept was adapted to money by Hal Finney in 2004 through the idea of “reusable proof of work.” [33] Following its introduction in 2009, bitcoin became the first widely adopted application of Finney's idea (Finney was also the recipient of the first bitcoin transaction). Proof of work forms the basis of many other cryptocurrencies as well.

Source: Proof of Work

[33] https://nakamotoinstitute.org/finney/rpow/index.html

Protocol

Return to Glossary

A **Protocol** is a set of rules and guidelines for communicating data. Rules are defined for each step and process during communication between two or more computers. Networks have to follow these rules to successfully transmit data.

Source: Protocol

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:protocol

Last update: **2020/11/13 02:58**

Protocol Layer

Return to Glossary

Protocol Layer, in Decentralized Finance (DeFi), are the Software protocols which are standards and rules written to govern specific tasks or activities. In parallel with real-world institutions, this would be a set of principles and rules that all participants in a given industry have agreed to follow as a prerequisite to operating in the industry. DeFi protocols are interoperable, meaning they can be used by multiple entities at the same time to build a service or an app. The protocol layer provides liquidity to the DeFi ecosystem. One example of a DeFi protocol is Synthetix, a derivatives trading protocol on Ethereum. It is used to create synthetic versions of real-world assets.[34]

Source: https://www.investopedia.com/decentralized-finance-defi-5113835

[34] Rakesh Sharma, Investopedia, 24 March 2021, Decentralized Finance (DeFi) Definition, Accessed 24 May 2021, https://www.investopedia.com/decentralized-finance-defi-5113835

Provenance

Return to Glossary

Provenance is meta-data about a record of the transformation of data such as inputs, entities, systems, and processes that influence data of interest. In other words, provenance is about the “how” and “what” of transfer, and transformation of data.

Source: OMG

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:provenance

Last update: **2020/11/13 13:16**

Public Key Infrastructure (PKI)

Return to Glossary

A **Public Key Infrastructure (PKI)** supports the distribution, revocation and verification of public keys used for public key encryption, and enables linking of identities with public key certificates. A PKI enables users and systems to securely exchange data over the internet and verify the legitimacy of certificate-holding entities, such as webservers, other authenticated servers and individuals. The PKI enables users to authenticate digital certificate holders, as well as to mediate the process of certificate revocation, using cryptographic algorithms to secure the process.

PKI certificates include a public key used for encryption and cryptographic authentication of data sent to or from the entity that was issued the certificate. Other information included in a PKI certificate includes identifying information about the certificate holder, about the PKI that issued the certificate, and other data including the certificate's creation date and validity period.

Source: https://searchsecurity.techtarget.com/definition/PKI

Public Network

Return to Glossary

A **Public Network** is completely open and anyone is free to join and participate in the core activities of the blockchain network. Anyone can join or leave, read, write and audit the ongoing activities on the public blockchain network, which helps a public blockchain maintain its self-governed nature. [35]

[35] “Public, Private, Permissioned Blockchains Compared”, Shobhit Seth, Investopedia, 10 April 2018, https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

Publish-Subscribe

Return to Glossary

Data Distribution Service (DDS) is a publish and subscribe service. Data values (Samples) are transferred through the system for conceptual “Data Objects”. The “publication” (the association of a Publisher and a Data Writer) send Samples to one or more “subscription” (the association of a Data Reader and a Subscriber).

Source: OpenSplice Glossary

Publisher

Return to Glossary

An entity created by a Domain Participant to manage a group of Data Writers. In order to associate with a Topic and publish samples of said Topic, the Publisher must be in the same DDS Domain and Partition and have a compatible set of Quality of Service (QoS) Policies.

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:p:publisher

Last update: **2021/06/15 14:23**

Q

Return to Glossary

Create a Glossary entry starting with 'Q' **Word or Expression** →	□ Add page

- Quality of Service (QoS) Policies

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:q:start

Last update: **2021/06/14 21:38**

Quality of Service (QoS) Policies

Return to Glossary

A rich set of characteristics that define the behaviour of the Data Distribution Service (DDS) systems (such as reliability, liveliness, durability, etc.). QoS Policies control the flow of the data through the system. The Topic, Data Reader, Data Writer, Publisher and Subscriber all have QoS policies. The QoS policies of Publisher, DataWriter, and Topic control the data on the sending side. QoS policies of Subscriber, Data Reader, and Topic control the data on the receiving side. These must be of a compatible type for successful communication, i.e. a Publisher/DW with a BEST_EFFORT QoS cannot send samples for a Topic with a RELIABLE QoS as it could degrade the Topic. However, a RELIABLE Publisher/DW can send samples for a BEST_EFFORT Topic.

Source: OpenSplice Glossary

R

Return to Glossary

Create a Glossary entry starting with 'R' **Word or Expression →**	▭ Add page

- Radio Frequency Identification (RFID)
- Random access memory (RAM)
- Read-Only Memory (ROM)
- Reboot the World Problem
- reCAPTCHA
- Recoverability
- Reduced Instruction Set Computer (RISC)
- Reference Architecture (RA)
- Referential Integrity
- Registered Agent
- Regression Testing
- Relational DataBase Management System (RDBMS)
- Relational Model (RM)
- Reliability Measure
- Reliability, Maintainability, and Availability (RAM)
- Relocatable Object
- Remote Procedure Call (RPC)
- Repairability
- Repeater
- Replaceability
- Representational State Transfer (REST)
- Request For Comment (RFC)
- Request For Information (RFI)
- Request For Proposal (RFP)
- Requirement
- RESTful API
- Reusability
- Rich Site Summary (RSS)
- Risk
- Roundoff Error
- Router

Radio Frequency Identification (RFID)

Return to Glossary

Radio Frequency Identification (RFID) refers to technologies that use wireless communication between an object (or tag) and interrogating device (or reader) to automatically track and identify such objects. The tag transmission range is limited to several meters from the reader. A clear line of sight between the reader and tag is not necessarily required.

Several industry groups, including the International Standards Organization (ISO) and International Electrotechnical Commission (IEC), regulate and define RFID interoperability standards. Source: https://www.techopedia.com/definition/3643/radio-frequency-identification-rfid

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:rfid

Last update: **2021/06/15 14:28**

Read-Only Memory (ROM)

Return to Glossary

Read-Only Memory (ROM) is a type of storage medium that permanently stores data on personal computers (PCs) and other electronic devices. It contains the programming needed to start a PC, which is essential for boot-up; it performs major input/output tasks and holds programs or software instructions.

Because ROM is read-only, it cannot be changed; it is permanent and non-volatile, meaning it also holds its memory even when power is removed. By contrast, Random access memory (RAM) is volatile; it is lost when power is removed.

Source: Read-Only Memory (ROM)

Reboot the World Problem

Return to Glossary

Reboot the World Problem occurs when an integral, underlying component of a system (usually middleware) changes requiring all the components that rely on that component to require a reboot in order to remain operational and interact with the other distributed components within the system. This is a cause of fragility in the overall system and is a major problem for Mission Critical System and Safety-Critical System (SCS) and risk the prime purpoose for the systems.

Source: local

reCAPTCHA

Return to Glossary

reCAPTCHA is a free service from Google that helps protect websites from spam and abuse. A “CAPTCHA” is a turing test to tell human and bots apart. It is easy for humans to solve, but hard for “bots” and other malicious software to figure out. By adding reCAPTCHA to a site, you can block automated software while helping your welcome users to enter with ease.

Source: https://support.google.com/recaptcha/?hl=en

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:recaptcha

Last update: **2020/11/15 02:11**

Recoverability

Return to Glossary

See 4.2.2.4 Recoverability

Recoverability is the ability of a system to be rebuilt in the event of a system failure do to human or natural disasters or catastrophic failures in hardware or software. After the system is recovered it is able to resume with full functionality with minimum interruption.

Source: local

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:recoverability

Last update: **2020/11/14 17:55**

Reduced Instruction Set Computer (RISC)

Return to Glossary

Reduced Instruction Set Computer (RISC) is a computer that uses a Central Processing Unit (CPU) that implements the processor design principle of simplified instructions. To date, RISC is the most efficient CPU architecture technology.

This architecture is an evolution and alternative to Complex Instruction Set Computer (CISC). With RISC, the basic concept is to have simple instructions that do less but execute very quickly to provide better performance.

Source: Reduced Instruction Set Computer (RISC)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:risc

Last update: **2020/11/16 15:30**

Reference Architecture (RA)

Return to Glossary

A **Reference Architecture** (RA) is defined as a set of goals:

- Provide common language for the various stakeholders
- Provide consistency of implementation of technology to solve problems
- Support the validation and comparison of implementations
- Encourage adherence to common standards, specifications, and patterns

Source: G. Doyle and B. Wilezynski, “Reference Architecture Description,” U. S. DoD, Washington, DC, 2010., Office of the Assistant Secretary of Defense for Networks and Information Integration (OASD/NII) Reference Architecture Description

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:reference_architecture

Last update: **2020/11/13 15:29**

Referential Integrity

Return to Glossary

Referential Integrity refers to the series of processes that make sure data is stored and used uniformly. Rules embedded into the database's structure about how foreign keys are used ensure that only appropriate changes, additions, or deletions of data occur. Rules may include constraints that eliminate the entry of duplicate data, guarantee that data is accurate, and/or disallow the entry of data that doesn't apply.

Source: Referential Integrity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:refintegrity

Last update: **2020/11/15 21:14**

Registered Agent

Return to Glossary

Registered Agent is a responsible third-party who is located in the same state in which a business entity was established and who is designated to receive service of process notices, correspondence from the Secretary of State, and other official government notifications, usually tax forms and notice of lawsuits, on behalf of the corporation or LLC.

Source: Registered Agent

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:registered_agent

Last update: **2020/11/13 15:29**

Regression Testing

Return to Glossary

Regression Testing is re-running tests for Functional Requirements and Non-Functional Requirements to ensure that previously developed and tested software still performs after a change. If not, that would be called a **regression** as far as functionality. Changes that may require regression testing include bug fixes, software enhancements, configuration changes, and even substitution of electronic components. As regression test suites tend to grow with each found defect, test automation is frequently involved. Sometimes a change impact analysis is performed to determine an appropriate subset of tests (non-regression analysis).

Source: https://en.wikipedia.org/wiki/Regression_testing

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:regressiontesting

Last update: **2020/12/20 21:00**

Relational DataBase Management System (RDBMS)

Return to Glossary

Relational DataBase Management System (RDBMS) is a database engine/system based on the relational model specified by Edgar F. Codd–the father of modern relational database design–in 1970.

Most modern commercial and open-source database applications are relational in nature. The most important relational database features include an ability to use tables for data storage while maintaining and enforcing certain data relationships.

Source: Relational DataBase Management System (RDBMS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:rdbms

Last update: **2020/11/13 15:29**

Reliability, Maintainability, and Availability (RAM)

Return to Glossary

Reliability, **Maintainability**, **and** **Availability** are three system attributes that are of great interest to systems engineers, logisticians, and users. Collectively, they affect both the utility and the life-cycle costs of a product or system.

Source: Reliability, Maintainability, and Availability (RAM)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:ram

Last update: **2020/11/13 21:56**

Reliability Measure

Return to Glossary

The **Reliability Measure** the risk of potential application failures and the stability of an application when confronted with unexpected conditions. According to ISO/IEC/IEEE 24765, Reliability is the degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time. The reason for checking and monitoring Reliability is to prevent or at least reduce application downtime, outages, data corruption, and errors that directly affect users.

Source: OASIS Security Services (SAML) TC - May 2012

Relocatable Object

Return to Glossary

A **Relocatable Object** file holds sections containing code and data. This file is suitable to be linked with other relocatable object files to create dynamic executable files, shared object files, or another relocatable object. A dynamic executable file holds a program that is ready to execute.

- Unix and Linux generally use Shared Object (.so)
- Microsoft Windows use Dynamic Link Library (.dll)

Source:
https://docs.oracle.com/cd/E23824_01/html/819-0690/chapter6-46512.html#:~:text=A%20relocatable%20object%20file%20holds,that%20is%20ready%20to%20execute.

Repairability

Return to Glossary

Repairability is the ability of a damaged or failed equipment, machine or system to be restored to acceptable operating condition within a specified period (repair time).

Source: Repairability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:repairability

Last update: **2020/11/14 18:21**

Repeater

Return to Glossary

A **Repeater** is an electronic device that amplifies the signal it receives. You can think of repeater as a device which receives a signal and retransmits it at a higher level or higher power so that the signal can cover longer distances, more than 100 meters for standard LAN cables. Repeaters work on the Physical layer.

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

Replaceability

Return to Glossary

See 2.2.1.3 Replaceability

Replaceability is the degree to which a product can replace another specified software product for the same purpose in the same environment.

Source: ISO 9126: 2001, 6.6.4

Representational State Transfer (REST)

Return to Glossary

Representational State Transfer (REST) is a distributed system framework that uses Web protocols and technologies. The REST architecture involves client and server interactions built around the transfer of resources. The Web is the largest REST implementation. Systems that conform to REST principles are referred to as RESTful.

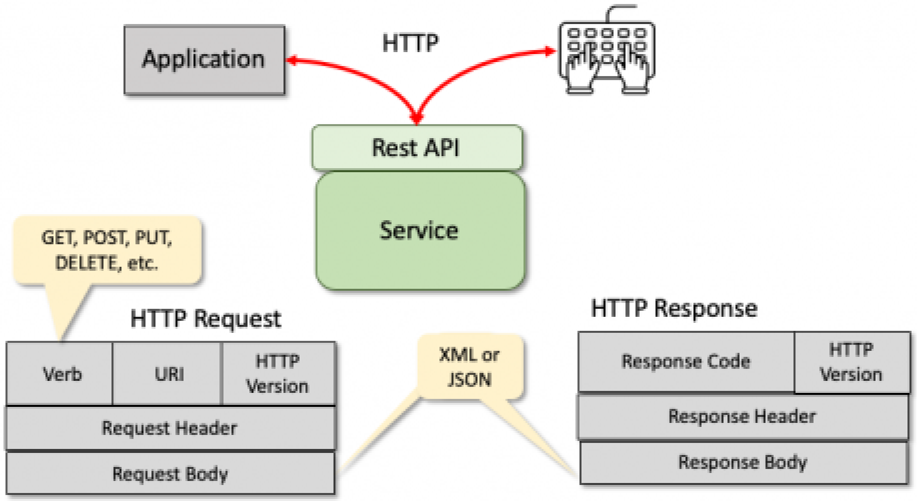


Figure 8: REST over HTTP

Source: Representational state transfer (REST)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:rest

Last update: **2021/05/01 21:41**

Request For Comment (RFC)

Return to Glossary

OMG

Request For Comment (RFC) is an alternative to the Technology Adoption Process allowing an OMG member to request OMG adoption of an uncontentious specification without requiring a Request for Proposals to be issued.

Source: OMG Request For Comment (RFC)

IETF

A **Request for Comments (RFC)** is a formal document drafted by the Internet Engineering Task Force (IETF) that describes the specifications for a particular technology. When an RFC is ratified, it becomes a formal standards document. RFCs were first used during the creation of the ARPANET protocols that came to establish what became today's Internet. They continue to be issued on an ongoing basis as the technology underlying the Internet evolves.

Source: IETF Request For Comment (RFC)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:rfc

Last update: **2020/11/13 15:29**

Request For Information (RFI)

Return to Glossary

Request For Information (RFI) A general request to the computer industry, academia, and any other interested parties to submit information about a particular technology area to one of the OMG's TFs. Information received in response to an RFI is typically used by a TF to formulate one or more RFPs.

Source: Request For Information (RFI)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:rfi

Last update: **2020/11/13 15:29**

Request For Proposal (RFP)

Return to Glossary

Request For Proposal (RFP) is the requirements document for a new OMG technology specification. Issuance of an RFP starts the OMG Technology Adoption Process. RFPs are written and recommended by a TF, certified by the AB, and issued by vote of the TF's Parent Body.

Source: Request For Proposals (RFP)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:rfp

Last update: **2020/11/13 15:29**

Requirement

Return to Glossary

A **Requirement** specifies a capability or condition that must (or should) be satisfied. A requirement may specify a function that a system must perform or a performance condition a system must achieve.

Source: Requirement

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:requirement

Last update: **2020/11/13 15:29**

RESTful API

Return to Glossary

RESTful API is an API that conforms to the representational state transfer or REST model. RESTful APIs are sometimes easier for developers to use because they have a familiar syntax and set of protocols. As more functionality has been built into the internet, developers have talked a lot about the benefits of RESTful architecture.

Source: RESTful API

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:restful

Last update: **2020/11/13 15:29**

Reusability

Return to Glossary

See 4.2.3.2 Reusability

Reusability is the use of existing assets in some form within the software product development process; these assets are products and by-products of the software development life cycle and include code, software components, test suites, designs and documentation. The opposite concept of Reusability is **leverage**, which modifies existing assets as needed to meet specific system requirements. Because reuse implies the creation of a separately maintained version of the assets[clarification needed], it is preferred over leverage.

Source: https://en.wikipedia.org/wiki/Reusability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:reusability

Last update: **2020/11/14 18:39**

Rich Site Summary (RSS)

Return to Glossary

Rss a web feed that allows users and applications to access updates to websites in a standardized, computer-readable format. These feeds can, for example, allow a user to keep track of many different websites in a single news aggregator. The news aggregator automatically checks the RSS feed for new content, allowing the list to be automatically passed from website to website or from website to user.

Note: Also known as Really Simple Syndication (RSS).

Source: Rich Site Summary (RSS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:rss

Last update: **2020/11/13 15:29**

Risk

Return to Glossary

Risk is a probability or threat of damage, injury, liability, loss, or any other negative occurrence that is caused by external or internal vulnerabilities, and that may be avoided through preemptive action.

Source: Risk

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:risk

Last update: **2020/11/13 15:29**

Roundoff Error

Return to Glossary

Roundoff Error is the difference between an approximation of a number used in computation and its exact (correct) value. In certain types of computation, roundoff error can be magnified as any initial errors are carried through one or more intermediate steps.

An egregious example of roundoff error is provided by a short-lived index devised at the Vancouver stock exchange (McCullough and Vinod 1999). At its inception in 1982, the index was given a value of 1000.000. After 22 months of recomputing the index and truncating to three decimal places at each change in market value, the index stood at 524.881, despite the fact that its “true” value should have been 1009.811.

Other sorts of roundoff error can also occur. A notorious example is the fate of the Ariane rocket launched on June 4, 1996 (European Space Agency 1996). In the 37th second of flight, the inertial reference system attempted to convert a 64-bit floating-point number to a 16-bit number, but instead triggered an overflow error which was interpreted by the guidance system as flight data, causing the rocket to veer off course and be destroyed.

Source: https://mathworld.wolfram.com/RoundoffError.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:roundofferror

Last update: **2021/03/13 04:47**

Router

Return to Glossary

A **Router** helps transmit packets to their destinations by charting a path through the sea of interconnected networking devices using different network topologies. Routers are intelligent devices, and they store information about the networks they're connected to. Most routers can be configured to operate as packet-filtering firewalls and use Access Control List (ACL). Routers, in conjunction with a channel service unit/data service unit (CSU/DSU), are also used to translate from LAN framing to Wide Area Network (WAN) framing. This is needed because LANs and WANs use different network protocols. Such routers are known as border routers. They serve as the outside connection of a LAN to a WAN, and they operate at the border of your network.

Router are also used to divide internal networks into two or more subnetworks. Routers can also be connected internally to other routers, creating zones that operate independently. Routers establish communication by maintaining tables about destinations and local connections. A router contains information about the systems connected to it and where to send requests if the destination isn't known. Routers usually communicate routing and other information using one of three standard protocols: Routing Information Protocol (RIP), Border Gateway Protocol (BGP) or Open Shortest Path First (OSPF).

Routers are your first line of defense, and they must be configured to pass only traffic that is authorized by network administrators. The routes themselves can be configured as static or dynamic. If they are static, they can only be configured manually and stay that way until changed. If they are dynamic, they learn of other routers around them and use information about those routers to build their routing tables.

Routers are general-purpose devices that interconnect two or more heterogeneous networks. They are usually dedicated to special-purpose computers, with separate input and output network interfaces for each connected network. Because routers and gateways are the backbone of large computer networks like the internet, they have special features that give them the flexibility and the ability to cope with varying network addressing schemes and frame sizes through segmentation of big packets into smaller sizes that fit the new network components. Each router interface has its own Address Resolution Protocol (ARP) module, its own LAN address (network card address) and its own Internet Protocol address (IP address). The router, with the help of a routing table, has knowledge of routes a packet could take from its source to its destination. The routing table, like in the bridge and switch, grows dynamically. Upon receipt of a packet, the router removes the packet headers and trailers and analyzes the Internet Protocol (IP) header by determining the source and destination addresses and data type, and noting the arrival time. It also updates the router table with new addresses not already in the table. The IP header and arrival time information is entered in the routing table. Routers normally work at the Network layer of the Open Systems Interconnection (OSI) Model.

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:r:router

Last update: **2020/11/14 14:16**

S

Return to Glossary

Create a Glossary entry starting with 'S' **Word or Expression →**	▭ Add page

- Safety Assurance (SfA)
- Safety-Critical System (SCS)
- Salami Slicing
- Sample
- Sanity Testing
- Sarbanes-Oxley Act (SOX)
- Scalability
- Scaling Out
- Scaling Up
- Script
- Secure Shell (SSH)
- Secure Sockets Layer (SSL)
- Security Measure
- Semantic Web
- Semantics
- Sensor
- Sequence
- Sequenced Packet Exchange (SPX)
- Server
- Servicability
- Session Layer
- Settlement_layer
- Shall (Requirement)
- Shared Library
- Shared Object (.so)
- Shielding
- Should (Requirement)
- Sidechain
- Simple Payment Verification (SPV)
- Single-Factor Authentication (SFA)
- Six Sigma (6Sigma)
- Smart Card
- Smart Contract
- Smoke Testing
- Snapshot
- Soft Fork
- Software Adaptability
- Software as a Service (SaaS)

- Software Assurance (SwA)
- Software Firewall
- Software Library
- Solution Stack
- Source Code
- Special Interest Group (SIG)
- Special Rules
- Sprint
- Stakeholder
- Standards Developing Organization (SDO)
- Standards Organization
- Standards Organization
- Standing Rules
- Static Library
- Statute
- Storage Device
- Straight-through Processing (StP)
- Stream Control Transmission Protocol (SCTP)
- Structured Query Language (SQL)
- Sub-Claim
- Subject Matter Expert (SME)
- Subscriber
- Subscriber Identity Module (SIM)
- Supervisory Control and Data Acquisition (SCADA)
- Supervisory Level
- Supply Chain
- Switch
- Symmetric Multiprocessing (SMP)
- Syntax
- System Assurance (SysA)
- System Lifecycle
- Systems and software Quality Requirements and Evaluation (SQuaRE)

Safety-Critical System (SCS)

Return to Glossary

A **Safety-Critical System (SCS)** or **Life-Critical System** is a system whose failure or malfunction may result in one (or more) of the following outcomes:

- death or serious injury to people
- loss or severe damage to equipment/property
- environmental harm

Source: Safety-Critical System (SCS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:safetycritical

Last update: **2020/11/13 17:34**

Safety Assurance (SfA)

Return to Glossary

Safety Assurance (SfA) is providing confidence that acceptable risk for the safety of personnel, equipment, facilities, and the public during and from the performance of operations is being achieved.

Source: FAA/NASA

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:safety_assurance

Last update: **2020/11/13 14:02**

Salami Slicing

Return to Glossary

Salami Slicing refers to a series of many small actions, often performed by clandestine means, that as an accumulated whole produces a much larger action or result that would be difficult or unlawful to perform all at once. The term is typically used pejoratively. Although salami slicing is often used to carry out illegal activities, it is only a strategy for gaining an advantage over time by accumulating it in small increments, so it can be used in legal ways as well.

An example of salami slicing, also known as penny shaving, is the fraudulent practice of stealing money repeatedly in extremely small quantities, usually by taking advantage of rounding to the nearest cent (or other monetary unit) in financial transactions. It would be done by always rounding down, and putting the fractions of a cent into another account. The idea is to make the change small enough that any single transaction will go undetected

Source: Salami Slicing

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:salami_slicing

Last update: **2020/11/13 14:02**

Sample

Return to Glossary

Sample is …

Source: URI

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sample

Last update: **2021/06/15 14:31**

Sanity Testing

Return to Glossary

Sanity Testing is a subset of regression testing. Sanity testing is performed to ensure that the code changes that are made are working as properly. Sanity testing is a stoppage to check whether testing for the build can proceed or not. The focus of the team during sanity testing process is to validate the functionality of the application and not detailed testing. Sanity testing is generally performed on build where the production deployment is required immediately like a critical bug fix.

Source: https://www.geeksforgeeks.org/sanity-testing-software-testing/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sanitytesting

Last update: **2020/12/20 20:59**

Sarbanes-Oxley Act (SOX)

Return to Glossary

The **Sarbanes-Oxley Act** (SOX) was designed to improve the quality of financial reporting by public companies. It was written in response to the fraudulent reporting of Enron Corporation, Worldcom, and several other businesses, and was passed in 2002. Key provisions of the Act are as follows:

- The CEO and CFO must certify the accuracy of the financial statements (Section 302).
- It is illegal to improperly influence how an audit is conducted (Section 303).
- Material off-balance sheet items must be disclosed (Section 401).
- Management must establish internal controls and report on their scope and accuracy, while the company's auditors must certify the reliability of those controls (Section 404).
- Substantial fines are imposed on anyone falsifying, stealing, or destroying records (section 802).
- Provides for the protection of whistleblowers from retaliation (Section 806).
- Sets criminal penalties when corporate officers do not certify the accuracy of the financial statements (Section 906).

The provisions of the Act made it significantly more expensive for firms to be publicly-held. The result was a decline in the number of public companies, especially among the smaller firms that could no longer afford the regulatory costs associated with being publicly-held. In particular, the requirements of Section 404 were considered to have the largest impact on the cost increase.

The official name of the Sarbanes-Oxley Act is the Corporate Responsibility Act of 2002.

Source: Sarbanes-Oxley Act

Scalability

Return to Glossary

See 4.2.9 Scalability

Scalability is an attribute that describes the ability of a process, network, software or organization to grow and manage increased demand. A system, business or software that is described as scalable has an advantage because it is more adaptable to the changing needs or demands of its users or clients.

Scalability is often a sign of stability and competitiveness, as it means the network, system, software or organization is ready to handle the influx of demand, increased productivity, trends, changing needs and even presence or introduction of new competitors.

Source: https://www.techopedia.com/definition/9269/scalability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:scalable

Last update: **2020/11/13 14:02**

Scaling Out

Return to Glossary

Scaling Out takes the infrastructure you've got, and replicates it to work in parallel. This has the effect of increasing infrastructure capacity roughly linearly. Data centers often scale out using pods. Build a compute pod, spin up applications to use it, then scale out by building another pod to add capacity. Actual application performance may not be linear, as application architectures must be written to work effectively in a scale-out environment.

Also see: Scaling Up

Source: https://packetpushers.net/scale-up-vs-scale-out/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:scaleout

Last update: **2020/11/16 21:20**

Scaling Up

Return to Glossary

Scaling Up is taking what you've got, and replacing it with something more powerful. From a networking perspective, this could be taking a 1GbE switch, and replacing it with a 10GbE switch. Same number of switchports, but the bandwidth has been scaled up via bigger pipes. The 1GbE bottleneck has been relieved by the 10GbE replacement.

Also see: Scaling Out

Source: https://packetpushers.net/scale-up-vs-scale-out/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:scaleup

Last update: **2020/11/16 21:18**

Script

Return to Glossary

A **Script** is a list of commands executed by certain programs or scripting engines. They are usually text documents with instructions written using a scripting language. They are used to generate Web pages and to automate computer processes.

Source: https://www.techopedia.com/definition/10324/scripts

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:script

Last update: **2021/06/15 14:31**

Secure Shell (SSH)

Return to Glossary

Secure Shell (SSH) is a cryptographic network Protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

SSH provides a secure channel over an unsecured network by using a Client-Server architecture, connecting an SSH Client application with an SSH Server. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. SSH is generally used to access Unix-like operating systems, but it can also be used on Microsoft Windows. Windows 10 uses OpenSSH as its default SSH client and SSH server.

Despite popular misconception, SSH is not an implementation of Telnet it is a replacement with cryptography provided by the Secure Sockets Layer (SSL).

Source: https://en.wikipedia.org/wiki/SSH_(Secure_Shell)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:ssh

Last update: **2021/06/15 14:32**

Secure Sockets Layer (SSL)

Return to Glossary

Secure Sockets Layer (SSL) is a standard Protocol used for the secure transmission of documents over a network. Developed by Netscape, SSL technology creates a secure link between a Web Server and browser (i.e., Client to ensure private and integral data transmission. SSL uses Transmission Control Protocol (TCP) for communication.

- **Note:** SSL is deprecated in favor of Transport layer security (TLS)

See: RFC6101 - The Secure Sockets Layer (SSL) Protocol Version 3.0

Source: https://www.techopedia.com/definition/24025/secure-sockets-layer-ssl

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:ssl

Last update: **2021/06/15 14:32**

Security Measure

Return to Glossary

The **Security Measure** assesses the degree to which an application protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization (ISO 25010). Security measures the risk of potential security breaches due to poor coding and architectural practices. Security problems have been studied extensively by the Software Assurance community and have been codified in the Common Weakness Enumeration (CWE).

Source: OASIS eXtensible Access Control Markup Language (XACML) TC - 22 January 2013

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:security_measure

Last update: **2020/11/13 14:02**

Semantics

Return to Glossary

Semantics differentiates the meaning of an instruction from its format. The format, which covers the spelling of language components and the rules controlling how components are combined, is called the language's syntax. For example, if you misspell a command, it is a syntax error. If, on the other hand, you enter a legal command that does not make any sense in the current context, it is a semantic error.

Source: Semantics

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:semantics

Last update: **2020/11/13 14:02**

Semantic Web

Return to Glossary

The Semantic Web is a mesh of data associated in such a way they can easily be processed by machines instead of human operators. It can be conceived as an extended version of the existing World Wide Web (WWW), and it represents an effective means of data representation in the form of a globally linked database. By supporting the inclusion of semantic content in Web pages, the Semantic Web targets the conversion of the presently available Web of unstructured documents to a Web of information/data.

Source: Semantic Web

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:semantic_web

Last update: **2020/11/13 14:03**

Sensor

Return to Glossary

A **Sensor** is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena. The output is generally a signal that is converted to human-readable display at the sensor location or transmitted electronically over a network for reading or further processing.

Source: Sensor

Sequence

Return to Glossary

A **Sequence** is logically composed of three things: an array of elements, a maximum number of elements that the array may contain (i.e. its allocated size), and a logical length indicating how many of the allocated elements are valid. The length may vary dynamically between 0 and the maximum (inclusive); it is not permissible to access an element at an index greater than or equal to the length.

A **Sequence** may either "own" the memory associated with it, or it may "borrow" that memory. If a **Sequence** owns its own memory, then the sequence itself will allocate the its memory and is permitted to grow and shrink that memory (i.e. change its maximum) dynamically.

Source: Sequence

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sequence

Last update: **2020/11/13 14:02**

Server

Return to Glossary

A **Server** is a computer, a device or a program that is dedicated to managing network resources. They are called that because they “serve” another computer, device, or program called “client” to which they provide functionality.

There are a number of categories of servers, including print servers, file servers, network servers and database servers. In theory, whenever computers share resources with client machines they are considered servers.

However, servers are often referred to as dedicated because they carry out hardly any other tasks apart from their server tasks.

Source: Server

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:server

Last update: **2020/11/13 14:03**

Servicability

Return to Glossary

Servicability is Degree to which the servicing of an item can be accomplished with given resources and within a specified timeframe.

Source: Servicability

Session Layer

Return to Glossary

The **Session Layer** is the fifth layer in the Open Systems Interconnection (OSI) Model, which controls the connections between multiple computers. The session layer tracks the dialogs between computers, which are also called sessions. This layer establishes, controls and ends the sessions between local and remote applications.

Source: Session Layer

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sessionlayer

Last update: **2020/11/13 14:03**

Settlement_layer

Return to Glossary

The **Settlement Layer**, in Decentralized Finance (DeFi), is also referred to as **Layer 0** because it is the base layer upon which other DeFi transactions are built. It consists of a public blockchain and its native digital currency or cryptocurrency. Transactions occurring on DeFi apps are settled using this currency, which may or may not be traded in public markets. One example of the settlement layer is Ethereum and its native token ether (ETH), which is traded at crypto exchanges. The settlement layer can also have tokenized versions of assets, such as the U.S. dollar, or tokens that are digital representations of real-world assets. For example, a real estate token might represent ownership of a parcel of land.[36]

Source: https://www.investopedia.com/decentralized-finance-defi-5113835

[36] Rakesh Sharma, Investopedia, 24 March 2021, Decentralized Finance (DeFi) Definition, Accessed 24 May 2021, https://www.investopedia.com/decentralized-finance-defi-5113835

Shall (Requirement)

Return to Glossary

Shall Requirement - Shall is used to indicate a requirement that is contractually binding, meaning it must be implemented, and its implementation verified. Period! Don’t think of “shall” as a word, but rather as an icon that SCREAMS: “This is a requirement.” If a statement does not contain the word “shall” it is not a requirement.

Source: https://reqexperts.com/2012/10/09/using-the-correct-terms-shall-will-should/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:shall_req

Last update: **2021/06/15 14:36**

Shared Library

Return to Glossary

Shared library are stored as `.so` (or in Windows `.dll`, or in OS X `.dylib`) files.

These are linked dynamically simply including the address of the library (whereas static linking is a waste of space). Dynamic linking links the libraries at the run-time. Thus, all the functions are in a special place in memory space, and every program can access them, without having multiple copies of them.

See: Static Library

Source: https://www.geeksforgeeks.org/difference-between-static-and-shared-libraries/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:shared_libraries

Last update: **2021/06/15 14:36**

Shared Object (.so)

Return to Glossary

A **Shared Object (.so)** is an indivisible unit that is generated from one or more Relocatable Objects. Shared objects can be bound with dynamic executables to form a runable process. As their name implies, shared objects can be shared by more than one application.

See: Shared Library and Static Library

Source:
https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/6n33n7f8u/index.html#:~:text=A%20shared%20object%20is%20an,by%20more%20than%20one%20application.

Shielding

Return to Glossary

Shielding is when Ethernet cables or shield twisted pair (STP) cables have an outside layer or “shield” of conductive material around the internal conductors, which needs to be grounded to cancel the effect of ElectroMagnetic Interference (EMI). The conductive shield can reflect or conduct external interference away without affecting the signals of the internal conductor. Therefore, shielded Ethernet cables are usually used to protect signals from EMI over the length of the cable run, so as to result in faster transmission speeds and fewer data errors.

Source: http://www.fiber-optic-components.com/shielded-vs-unshielded-ethernet-cable-use.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:shielding

Last update: **2021/06/15 14:37**

Should (Requirement)

Return to Glossary

Should is not per se the specification of a requirement but is used to capture Goals, non-mandatory provisions. Should is used to indicate a goal which must be addressed by the design team but is not formally verified.

Why include should (goal) statements in your requirement document? Because you may have a very important issue that you want to communicate to the developers, but can’t think of a way to do so in the form of a verifiable requirement. For example, NASA was developing a jet pack called SAFER and one of the requirements read “The SAFER shall not impede crew mobility”. Well anything but a decal will probably impeded crew mobility so how am I going to verify that statement if it is written as a requirement? I can’t. So, now that I recognize this, I change the statement from a requirement “shall” to a goal “should” and then I ask my designers and developers at every subsequent review, “how are you going to design the jet pack so that it does not impeded crew mobility”?

Source: https://reqexperts.com/2012/10/09/using-the-correct-terms-shall-will-should/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:should_req

Last update: **2021/06/15 14:38**

Sidechain

Return to Glossary

A **Sidechain** is a secondary Blockchain connected to the main blockchain with a two-way peg. Sidechains may have their own consensus protocols, which could be completely different from the mainchain's protocol. Theoretically, a sidechain can add new functionalities, improve privacy, and security of traditionally vanilla blockchains.

Source: https://www.sciencedirect.com/science/article/abs/pii/S1084804519303315

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sidechain

Last update: **2021/06/15 14:38**

Simple Payment Verification (SPV)

Return to Glossary

Simple Payment Verification (SPV) is a technique described in Satoshi Nakamoto's paper allowing a lightweight client to verify that a transaction is included in the Bitcoin blockchain, without downloading the entire blockchain. The client just needs to download the headers rather than the heavier full blocks.

Source: Simple Payment Verification (SPV)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:spv

Last update: **2020/11/13 14:03**

Single-Factor Authentication (SFA)

Return to Glossary

Single-Factor Authentication (SFA) is a process for securing access to a given system, such as a network or website, that identifies the party requesting access through only one category of credentials.

The most common example of SFA is password-based authentication. Password security relies on the diligence of the system administrator or user who sets up the account. Best practices include creating a strong password and ensuring that no one can access it.

Source: https://searchsecurity.techtarget.com/definition/single-factor-authentication-SFA

Six Sigma (6Sigma)

Return to Glossary

Six Sigma is a quality-control methodology developed in 1986 by Motorola, Inc. The method uses a data-driven review to limit mistakes or defects in and process. … Six Sigma points to the fact that, mathematically, it would take a six-standard-deviation event from the mean for an error to happen.

Source: https://www.investopedia.com/terms/s/six-sigma.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:6sigma

Last update: **2020/11/14 12:04**

Smart Card

Return to Glossary

A **Smart Card**, **Chip Card**, or **Integrated Circuit Card (ICC)** is a physical electronic authorization device, used to control access to a resource. It is typically a plastic credit card-sized card with an embedded integrated circuit (IC) chip.[1] Many smart cards include a pattern of metal contacts to electrically connect to the internal chip. Others are contactless, and some are both. Smart cards can provide personal identification, authentication, data storage, and application processing. Applications include identification, financial, mobile phones (SIM), public transit, computer security, schools, and healthcare. Smart cards may provide strong security authentication for single sign-on (SSO) within organizations. Numerous nations have deployed smart cards throughout their populations.

Source: https://en.wikipedia.org/wiki/Smart_card

Smart Contract

Return to Glossary

A **Smart Contract** is a decentralized application that executes business logic in response to events. Smart contract execution can result in the exchange of money, delivery of services, unlocking of content protected by Digital Rights management, or other types of data manipulation such as changing the name on a land title. Smart contracts can also be used to enforce privacy protection by, for example, facilitating the selective release of privacy-protected data to meet a specific request.

There are a variety of architectures for how the programs underpinning smart contracts are developed, distributed, managed, and updated. They can be stored as part of a blockchain or other distributed ledger technology, and integrated into various payment mechanisms and digital exchanges that can include bitcoin and other cryptocurrencies.

Despite the name, smart contracts are not legally binding contracts. Their main function is to programmatically execute business logic that performs various tasks, processes, or transactions that have been programmed into them to respond to a given set of conditions. Legal steps must be undertaken to link this execution to legally binding agreements between parties.

Source: https://searchcompliance.techtarget.com/definition/smart-contract

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:smart_contracts

Last update: **2021/06/11 16:07**

Smoke Testing

Return to Glossary

Smoke Testing is performed on the ‘new’ build given by developers to QA team to verify if the basic functionalities are working or not. It is one of the important functional testing types. This should be the first test to be done on any new build. In smoke testing, the test cases chosen cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionality of the system is working fine.

Source: https://www.simform.com/functional-testing-types/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:smoketesting

Last update: **2020/11/15 02:17**

Snapshot

Return to Glossary

Snapshot is done to prevent the Tangle (DAG) from expanding too much in size. Snapshotting saves all the balances, while removing the history and data of all the transactions to start fresh. These addresses with balances act like a new genesis address, but no previous history or data will be attached.

Source: Snapshot

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:snapshot

Last update: **2020/11/13 14:07**

Soft Fork

Return to Glossary

a soft fork (or sometimes softfork) is a change to the software protocol where only previously valid blocks/transactions are made invalid. Since old nodes will recognize the new blocks as valid, a softfork is backward-compatible. This kind of fork requires only a majority of the miners upgrading to enforce the new rules, as opposed to a hard fork which requires all nodes to upgrade and agree on the new version.

Source: Soft Fork

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:soft_fork

Last update: **2020/11/13 14:03**

Software Adaptability

Return to Glossary

Software Adaptability is when software components with a well-defined, stable Application Programming Interface (API) can be exchanged using another components with minimal effort as long as the component adheres to the API. For example, SQL describes an API for a database component. As long as the software adheres to the standard SQL API, the DataBase Management System (DBMS) can be exchanged from Oracle to PostreSQL with no to minimal impact.[37].

Source: https://personal.utdallas.edu/~chung/ftp/sqm.pdf

[37] Nary Subramanian and Lawrence Chung, Metrics for Software Adaptability, University of Texas - Dallas, Accessed 29 July 2020, https://personal.utdallas.edu/~chung/ftp/sqm.pdf

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:swadapt

Last update: **2020/11/13 14:02**

Software as a Service (SaaS)

Return to Glossary

Software as a Service (SaaS) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. SaaS is one of three main categories of cloud computing, alongside Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS).

Source: Software as a service (SaaS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:saas

Last update: **2020/11/13 15:29**

Software Assurance (SwA)

Return to Glossary

Software Assurance (SwA) is

1. The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software throughout the lifecycle (DoD);
2. The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures (NASA)

Source: Committee on National Security Systems, CNS 4009 Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:swassurance

Last update: **2020/11/14 20:56**

Software Firewall

Return to Glossary

A **Software Firewall** is a program, which runs in the background on a computer. Enterprise edition software firewalls allow IT professionals to manage firewall rules and configurations therefore taking the configuration and management responsibility away from the user.

The software firewall operates by inspecting the network data that is incoming to the computer, as well as outgoing from the computer. If the firewall detects the data has been altered or if appears malicious, the software blocks it, effectively preventing any form of an attack.

Firewalls operate from a set of rules. The administrator can make a specific rule in the firewall to block a specific website or even a specific application. In some cases when connecting to a printer with advanced features, the IT professional may need to alter the firewall rules to allow certain data through to access features.

Source: https://www.brickhost.com/software-firewall-vs-hardware-firewall/

Software Library

Return to Glossary

A **Software Library** is a suite of data and programming code that is used to develop software programs and applications. It is designed to assist both the programmer and the programming language Compiler in building and executing software.

Source: Software Library

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:swlib

Last update: **2020/11/13 14:00**

Solution Stack

Return to Glossary

A **Solution Stack** is an ordered collection of software that makes it possible to complete a particular task.

There are a lot of different types of solution stacks. Here are a few examples:

- A server stack includes the software required for basic server functioning.
- A Web stack includes the software required for Web app development.
- An application stack includes all the application programs required to perform a given task.
- A software stack includes the software required for a given task. (Software stacks include infrastructure software, rather than just applications.)
- A storage stack is a type of software stack that includes servers, networking components and server virtualization components.
- A virtualization stack is the collection of resources that, along with the hypervisor, make up the Microsoft Hyper-V environment.

Source: https://whatis.techtarget.com/definition/solution-stack

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:solutionstack

Last update: **2021/06/15 14:39**

Source Code

Return to Glossary

Source Code is the set of instructions and statements written by a programmer using a computer programming language (i.e., C/C++, Java, JavaScrit, Python, Erlang, etc.). This code is later translated into machine language by a Compiler. The translated code is referred to as object code.

Source: Source Code

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sourcecode

Last update: **2020/11/13 15:29**

Special Interest Group (SIG)

Return to Glossary

Special Interest Group (SIG) is one of the three types of OMG Subgroups, a SIG may be chartered by any one of the three Plenary Bodies. Unlike Task Forces (TFs), SIGs may not recommend issuance of RFPs or adoption of technology to their Parent Body, nor may they issue RFIs although they may issue Surveys which serve the same function as an RFI but carry a different name. Typically, SIGs function as discussion groups and/or technology incubators, i.e, primary authors of RFPs or RFIs. Occasionally a SIG will be disbanded and its Parent Body will charter a TF with a similar name and function, typically after a period of several months to a year or more during which the SIG has demonstrated that its membership could sustain the effort necessary to recommend and maintain adopted technology.

Source: Special Interest Group (SIG)

Special Rules

Return to Glossary

Special Rules are rules that supplement or modify the group's chosen parliamentary authority.

Source: Special Rules

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:special_rules

Last update: **2020/11/13 14:07**

Sprint

Return to Glossary

A **Sprint** is a short, time-boxed period when a scrum team works to complete a set amount of work. Sprints are at the very heart of scrum and agile methodologies, and getting sprints right will help your agile team ship better software with fewer headaches.

Source: Sprint

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sprint

Last update: **2020/11/16 18:16**

Stakeholder

Return to Glossary

Stakeholder is is a person, group or organization that has interest or concern in a *[target]* organization. Stakeholders can affect or be affected by the [target] organization's actions, objectives and policies. Note: *[target]* added for clarification purposes

Source: Stakeholder

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:stakeholder

Last update: **2020/11/13 14:00**

Standards Developing Organization (SDO)

Return to Glossary

Standards Developing Organization (SDO), Standards Organization, Standards Body, or Standards Setting Organization (SSO) is an organization whose primary activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting, or otherwise producing technical standards[1] that are intended to address the needs of a group of affected adopters.

Most standards are voluntary in the sense that they are offered for adoption by people or industry without being mandated in law. Some standards become mandatory when they are adopted by regulators as legal requirements in particular domains.

Source: Standards Developing Organization (SDO)

Standards Organization

Return to Glossary

Standards Organizations are organizations whose primary activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting, or otherwise producing technical standards that are intended to address the needs of a group of affected adopters. [38]

Source: Standards Organization

[38] Intercloud: Solving Interoperability and Communication in a Cloud of Clouds, Appendix A, by Ken Owens, Monique Morrow, Venkata Josyula, Jazib Frahim, Publisher: Cisco Press Release Date: June 2016 ISBN: 9780134189239,https://learning.oreilly.com/library/view/intercloud-solving-interoperability/9780134189239/app01.html

Standards Organization

Return to Glossary

Standards Organizations are organizations whose primary activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting, or otherwise producing technical standards that are intended to address the needs of a group of affected adopters. [39]

Source: Standards Organization

[39] Intercloud: Solving Interoperability and Communication in a Cloud of Clouds, Appendix A, by Ken Owens, Monique Morrow, Venkata Josyula, Jazib Frahim, Publisher: Cisco Press Release Date: June 2016 ISBN: 9780134189239,https://learning.oreilly.com/library/view/intercloud-solving-interoperability/9780134189239/app01.html

Standing Rules

Return to Glossary

Standing Rules are regulations or rules that deal with the procedures and operations of a business or guidance of an institution or administration of a society and are adopted from time to time similarly as any other act of the deliberative assembly. Generally, the standing rules can be amended only by a majority vote.

Source: Standing Rules

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:standing_rules

Last update: **2020/11/13 14:03**

Static Library

Return to Glossary

A **Static Library** or **Statically-Linked Library** is a set of routines, external functions and variables which are resolved in a caller at compile-time and copied into a target application by a compiler, linker, or binder, producing an object file and a stand-alone executable. This executable and the process of compiling it are both known as a static build of the program. Historically, libraries could only be static.

They are usually faster than the shared libraries because a set of commonly used object files is put into a single library executable file. One can build multiple executables without the need to recompile the file. Because it is a single file to be built, use of link commands are simpler than shared library link commands, because you specify the name of the static library.

See: Shared Library

Source: https://www.geeksforgeeks.org/difference-between-static-and-shared-libraries/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:static_library

Last update: **2021/06/15 14:39**

Statute

Return to Glossary

Statute is a formally drafted and written law adopted by both chambers or houses of a legislature. Statutes are enacted usually by voting following an open discussion, and signed thereafter by the head of State and included in the country's statute book.

Source: Statute

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:statute

Last update: **2020/11/13 14:03**

Storage Device

Return to Glossary

Storage Device is any computing hardware that is used for storing, porting and extracting data files and objects. It can hold and store information both temporarily and permanently, and can be internal or external to a computer, server or any similar computing device.

A storage device may also be known as a storage medium or storage media.

Source: Storage Device

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:storagedevice

Last update: **2020/11/14 17:58**

Straight-through Processing (StP)

Return to Glossary

Straight-through Processing is an automated electronic payment process that is used by corporations and banks. STP allows for the entire payment process, from initiation to final settlement, to be free of human intervention. Straight-through processing can help local businesses, as well as large corporations, pay and receive money faster than the traditional process.

Source: Straight-through Processing (StP)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:stp

Last update: **2020/11/13 15:29**

Structured Query Language (SQL)

Return to Glossary

Structured Query Language (SQL) is a programming language that is typically used in relational database or data stream management systems.

It was developed by IBM in the early 1970s and is now an official standard recognized by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO).

Source:
https://www.techopedia.com/definition/1245/structured-query-language-sql#:~:text=Structured%20Query%20Language%20(SQL)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sql

Last update: **2021/06/15 14:44**

Sub-Claim

Return to Glossary

Subclaim is a subordinate Claim

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4548534/#b13-v115.n03.a05

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:subclaim

Last update: **2020/11/13 15:29**

Subject Matter Expert (SME)

Return to Glossary

A **Subject Matter Expert (SME)** is an individual who is considered an expert on particular subjects, or flagged as an expert in a piece of management software or other technology. The subject matter expert has a particular territory in which he or she has demonstrated above-average knowledge or experience. Source: https://www.techopedia.com/definition/30084/subject-matter-expert-sme

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:sme

Last update: **2021/06/15 14:44**

Subscriber

Return to Glossary

An entity created by a Domain Participant to manage a group of Data Readers. In order to subscribe to a Topic the Subscriber must be in the same DDS Domain and Partition, and have a compatible set of Quality of Service (QoS) Policies associated with it.

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:subscriber

Last update: **2021/06/15 14:46**

Subscriber Identity Module (SIM)

Return to Glossary

Subscriber Identity Module (SIM) or **Subscriber Identification Module (SIM)**, widely known as a SIM card, is an integrated circuit that is intended to securely store the international mobile subscriber identity (IMSI) number and its related key, which are used to identify and authenticate subscribers on mobile telephony devices (such as mobile phones and computers). It is also possible to store contact information on many SIM cards. SIM cards are always used on GSM phones; for CDMA phones, they are needed only for LTE-capable handsets. SIM cards can also be used in satellite phones, smart watches, computers, or cameras.

Source: https://en.wikipedia.org/wiki/SIM_card

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:simcard

Last update: **2020/11/16 13:32**

Supervisory Control and Data Acquisition (SCADA)

Return to Glossary

Supervisory Control and Data Acquisition (SCADA) refers to industrial control systems (ICS) that are employed to control and keep track of equipment or a plant in industries like water and waste control, telecommunications, energy, transport, and oil and gas refining. SCADA is a computer system used to gather and analyze real-time data. This data is processed by the computer and is presented on a regular basis. SCADA also saves and make logs for every event into a log file that is saved on a hard drive or is sent to a printer. SCADA gives warnings by sounding alarms if situations develop into hazardous scenarios.

Source: Supervisory Control and Data Acquisition (SCADA)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:scada

Last update: **2020/11/15 20:19**

Supervisory Level

Return to Glossary

Supervisory Level is the Supervisory Control and Data Acquisition (SCADA) is combines the Field and Control Levels to provide oversight from a single location. This is usually accomplished using Graphical User Interface, or Human-machine interface (HMI), to remotely control operations. For example, water plants often employ this technology to control remote water pumps.

Source: Supervisory Level

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:superlevel

Last update: **2020/11/15 20:17**

Supply Chain

Return to Glossary

The **Supply Chain** comprises the flow of all information, products, materials, and funds between different stages of creating and selling a product to the end-user. The concept of the supply chain comes from an operational management perspective. Every step in the process—including creating a good or service, manufacturing it, transporting it to a place of sale, and selling it—is part of a company's supply chain.

The supply chain includes all functions involved in receiving and filling a customer request. These functions include:

- Product development
- Marketing
- Operations
- Distribution
- Finance
- Customer service

Source: [[https://www.investopedia.com/ask/answers/043015/what-difference-between-value-chain-and-supply-chain.asp]]

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:supplychain

Last update: **2021/06/15 14:47**

Switch

Return to Glossary

A **Switch** generally has a more intelligent role than a Hub. A switch is a multiport device that improves network efficiency. The switch maintains limited routing information about nodes in the internal network, and it allows connections to systems like hubs or routers. Strands of LANs are usually connected using switches. Generally, switches can read the hardware addresses of incoming packets to transmit them to the appropriate destination.

Using switches improves network efficiency over hubs or routers because of the virtual circuit capability. Switches also improve network security because the virtual circuits are more difficult to examine with network monitors. You can think of a switch as a device that has some of the best capabilities of routers and hubs combined. A switch can work at either the Data Link Layer (DLL) or the Network layer of the Open Systems Interconnection (OSI) Model. A multilayer switch is one that can operate at both layers, which means that it can operate as both a switch and a router. A multilayer switch is a high-performance device that supports the same routing protocols as routers.

Switches can be subject to Distributed Denial-of-Service (DDoS) attacks; flood guards are used to prevent malicious traffic from bringing the Switch to a halt. Switch port security is important so be sure to secure switches: Disable all unused ports and use Dynamic Host Configuration Protocol (DHCP) snooping, Address Resolution Protocol (ARP) Spoofing and MAC address filtering.

Source: https://blog.netwrix.com/2019/01/08/network-devices-explained/

Symmetric Multiprocessing (SMP)

Return to Glossary

Symmetric Multiprocessing (SMP) is the processing of programs by multiple processors that share a common operating system and memory. In symmetric (or “tightly coupled”) multiprocessing, the processors share memory and the I/O bus or data path. A single copy of the operating system is in charge of all the processors. SMP, also known as a “shared everything” system, does not usually exceed 16 processors.

SMP systems are considered better than MPP systems for online transaction processing (OTP) in which many users access the same database in a relatively simple set of transactions. An advantage of SMP for this purpose is the ability to dynamically balance the workload among computers (and as a result serve more users faster).[40]

Source:
https://searchdatacenter.techtarget.com/definition/SMP?_ga=2.233172572.392289681.1607438816-2133032675.1592258365

[40] Rouse, Margret; Definition of symmetric multiprocessing, WhatIs.com, Accessed 8 December 2020, https://searchdatacenter.techtarget.com/definition/SMP?_ga=2.233172572.392289681.1607438816-2133032675.1592258365

Syntax

Return to Glossary

Syntax refers to the rules that specify the correct combined sequence of symbols that can be used to form a correctly structured program using a given programming language. Programmers communicate with computers through the correctly structured syntax, semantics and grammar of a programming language.

Source: Syntax

System Assurance (SysA)

Return to Glossary

System Assurance (SysA) is the planned and systematic set of engineering activities necessary to assure that products conform with all applicable system requirements for safety, security, reliability, availability, maintainability, standards, procedures, and regulations, to provide the user with acceptable confidence that the system behaves as intended in the expected operational context.

Source: OMG SysA Task Force

System Lifecycle

Return to Glossary

The **System Lifecycle** in systems engineering is a view of a system or proposed system that addresses all phases of its existence to include system conception, design and development, production and/or construction, distribution, operation, maintenance and support, retirement, phase-out and disposal.

Source: System Lifecycle

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:s:syslifecycle

Last update: **2020/11/13 15:29**

Systems and software Quality Requirements and Evaluation (SQuaRE)

Return to Glossary

Systems and software Quality Requirements and Evaluation (SQuaRE) is an extension (ISO/IEC 25050 to ISO/IEC 25099) and is designated to contain system or software product quality International Standards and/or Technical Reports that address specific application domains or that can be used to complement one or more SQuaRE International Standards.

Source: ISO/IEC 25010:2011

T

Return to Glossary

Create a Glossary entry starting with 'T' **Word or Expression** →	□	Add page

- Tangle
- Taxonomy
- Technical Standard
- Testability
- Throughput
- Tokens
- Topic
- Total Cost of Ownership (TCO)
- Transmission Control Protocol (TCP)
- Transport Layer
- Transport layer security (TLS)
- Two-Factor Authentication (2FA)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:start

Last update: **2021/06/14 21:39**

Tangle

Return to Glossary

Tangle is the moniker used to describe IOTA’s Directed Acyclic Graph (DAG) based transaction settlement and data integrity layer focused on the Internet of Things (IOT). The Tangle is essentially a string of individual transactions that are interlinked to each other and stored through a decentralized network of node participants.

Importantly, the Tangle does not have miners as users of the network function as the miners themselves through performing small computational Proof of Work (PoW) for each transaction by verifying previous transactions submitted to the network. Focused on allowing the network to scale for a global micropayment network of interconnected IoT devices, The Tangle is designed to offer a solution to the heterogeneous nature of current blockchain systems.

Source: What is The Tangle? Complete Guide to IOTA’s Directed Acyclic Graph (DAG)

Taxonomy

Return to Glossary

Taxonomy formalizes the hierarchical relationships among concepts and specifies the term to be used to refer to each; it prescribes structure and terminology.

- A taxonomy is a form of classification scheme
- Taxonomies are semantic
- A taxonomy is a kind of knowledge map

Sources:

- Patrick Lambe Defining Taxonomy and
- Taxonomy

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:taxonomy

Last update: **2020/11/13 15:29**

Technical Standard

Return to Glossary

A **Technical Standard** is an established norm or requirement in regard to technical systems captured as formal document establishing uniform engineering or technical criteria, methods, processes, and practices. **Technical Standards** are defined by standards organizations.

Source: Technical Standard

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:technical_standard

Last update: **2020/11/13 15:29**

Testability

Return to Glossary

See 2.2.3.5 Testability

Testability is the degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/57-maintainability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:testability

Last update: **2020/11/15 02:25**

Throughput

Return to Glossary

Throughput refers to how much data can be transferred from one location to another in a given amount of time. It is used to measure the performance of hard drives and RAM, as well as Internet and network connections.

Source: Throughput

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:thruput

Last update: **2020/11/16 19:03**

Tokens

Return to Glossary

Tokens are digital assets, issued by the project, which can be used as a method of payment inside project's ecosystem, performing similar functions with Coins, but **the main difference is that it also gives the holder a right to participate in the network.** It may perform the functions of digital asset, represent a company's share, give access to the project's functional and many more—with the launching of new projects unknown facets of tokens' functional are discovered. Ticket to a concert, for example, is a "real-life token"—you may use it at a certain time, at a certain place. You can't go to the restaurant and pay your bill with concert ticket—ticket has its value only at concert hall. Digital tokens are the same—they have certain use case only inside certain project.

Tokens represent an asset or utility, so security and utility tokens are distinguished. Security tokens are designed to be the company's share (token of notorious project DAO, that was hacked right after launching, was recognized as security token), while utility tokens have certain use case inside the project (BON Token).

Creating a token is easier than creating a coin, as you don't have to create a new code or modify already existing one—you just use a standard template from platforms like Ethereum, that are blockchain-based and allow anyone to create tokens in just few steps. Using a template for creating tokens provides smooth interoperability, so users can store different types of tokens in one wallet. Ethereum was the first to simplify the process of creating a token, being not the last reason why tokens flooded the market.

Compare to Coins

Source: Tokens

Topic

Return to Glossary

A **Topic** is the most basic description of data that is to be published and/or subscribed to. A topic connects a Data Writer with a Data Reader, i.e., communication does not occur unless the topic published by a data writer matches a topic subscribed to by a data reader. Communication via topics is anonymous and transparent, i.e., publishers and subscribers need not be concerned with how topics are created or who is writing/reading them since the Data Distribution Service (DDS) DCPS middleware manages these issues. In programming terms, a Topic is a Class and each instance is a Sample. Topic instances are associated with a key (similar to a packet ID), defined by IDL and a set of Quality of Service parameters.

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:topic

Last update: **2021/06/15 15:29**

Total Cost of Ownership (TCO)

Return to Glossary

Total Cost of Ownership (TCO) is the purchase price of an asset plus the costs of operation. Assessing the total cost of ownership represents taking a bigger picture look at what the product is and what its value is over time.

When choosing among alternatives in a purchasing decision, buyers should look not just at an item's short-term price, known as its purchase price, but also at its long-term price, which is its total cost of ownership. The item with the lower total cost of ownership is the better value in the long run.

Source: https://www.investopedia.com/terms/t/totalcostofownership.asp

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:totalcostown

Last update: **2020/11/13 17:49**

Transmission Control Protocol (TCP)

Return to Glossary

Transmission Control Protocol (TCP) is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other.

Source: Transmission Control Protocol (TCP)

Transport Layer

Return to Glossary

The **Transport Layer** is seven-layer Open Systems Interconnection (OSI) Model of computer networking, the physical layer or layer 1 is the first and lowest layer. The implementation of this layer is often termed PHY.

The physical layer consists of the basic networking hardware transmission technologies of a network. It is a fundamental layer underlying the logical data structures of the higher level functions in a network. Due to the plethora of available hardware technologies with widely varying characteristics, this is perhaps the most complex layer in the OSI architecture.

The physical layer defines the means of transmitting raw bits rather than logical data packets over a physical link connecting network nodes. The bit stream may be grouped into code words or symbols and converted to a physical signal that is transmitted over a hardware transmission medium. The physical layer provides an electrical, mechanical, and procedural interface to the transmission medium. The shapes and properties of the electrical connectors, the frequencies to broadcast on, the modulation scheme to use and similar low-level parameters, are specified here.

Within the semantics of the OSI network architecture, the physical layer translates logical communications requests from the Data Link Layer (DLL) into hardware-specific operations to affect transmission or reception of electronic signals.

Source: Transport Layer

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:transportlayer

Last update: **2020/11/13 15:29**

Transport layer security (TLS)

Return to Glossary

Transport layer security (TLS) is a protocol that provides communication security between client/server applications that communicate with each other over the Internet. It enables privacy, integrity and protection for the data that's transmitted between different nodes on the Internet. TLS is a successor to the Secure Sockets Layer (SSL) protocol.

See: RFC2818 - HTTP Over TLS (HTTPS) See: RFC2246 - The TLS Protocol

Source: Transport layer security (TLS)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:tls

Last update: **2021/06/15 15:38**

Two-Factor Authentication (2FA)

Return to Glossary

Two-Factor Authentication (2FA), sometimes referred to as two-step verification or dual-factor authentication, is a security process in which users provide two different authentication factors to verify themselves. This process is done to better protect both the user's credentials and the resources the user can access. Two-factor authentication provides a higher level of security than authentication methods that depend on Single-Factor Authentication (SFA), in which the user provides only one factor – typically, a password or passcode. Two-factor authentication methods rely on a user providing a password, as well as a second factor, usually either a security token or a biometric factor, such as a fingerprint or facial scan.

Two-factor authentication adds an additional layer of security to the authentication process by making it harder for attackers to gain access to a person's devices or online accounts because knowing the victim's password alone is not enough to pass the authentication check. Two-factor authentication has long been used to control access to sensitive systems and data, and online service providers are increasingly using 2FA to protect their users' credentials from being used by hackers who have stolen a password database or used phishing campaigns to obtain user passwords.

Source: https://searchsecurity.techtarget.com/definition/two-factor-authentication

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:t:2fa

Last update: **2021/05/10 13:15**

U

Return to Glossary

Create a Glossary entry starting with 'U' **Word or Expression** →		Add page

- Unicode Transformation Format (UTF)
- Unified Modeling Language (UML)
- Uninterruptible Power Supply (UPS)
- Unique Identifier (UID)
- Unit Testing
- Universal Serial Bus (USB)
- Universally Unique IDentifier (UUID)
- UNIX
- UNIX Domain Scoket (Socket)
- Upload Speed
- Usability
- Use Case
- User Datagram Protocol (UDP)
- User Defined Integrity
- User Error Protection
- User Interface Aesthetics
- User Scenario

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:start

Last update: **2021/06/14 21:39**

Unicode Transformation Format (UTF)

Return to Glossary

The **Unicode Transformation Format (UTF)** is a character encoding format which is able to encode all of the possible character code points in Unicode. The most prolific is UTF-8, which is a variable-length encoding and uses 8-bit code units, designed for backwards compatibility with ASCII encoding.

Different kinds of UTF encodings include:

- **UTF-1** — Retired predecessor of UTF-8, no longer part of the Unicode Standard
- **UTF-7** — Uses 7 bits for encoding and was primarily used in email, but is now considered obsolete
- **UTF-8** — Uses an 8-bit variable-width encoding in order to maximize compatibility with ASCII
- **UTF-16** — 16-bit variable-width encoding
- **UTF-32** — 32-bit fixed-width encoding
- **UTF-EBCIDC** — Uses 8 bits and designed to be compatible with Extended Binary Coded Decimal Interchange Code (EBCDIC)

Source: https://www.techopedia.com/definition/976/unicode-transformation-format-utf

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:utf

Last update: **2021/06/15 15:38**

Unified Modeling Language (UML)

Return to Glossary

Unified Modeling Language (UML) is the OMG standard language for Analysis and Design of applications, specifying the structure and behavior of systems. UML is defined as an underlying abstract syntax and an overlying graphical concrete representation.

Source: Unified Modeling Language (UML)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:uml

Last update: **2020/11/13 13:11**

Uninterruptible Power Supply (UPS)

Return to Glossary

An **Uninterruptible Power Supply (UPS)** is an electrical apparatus that provides emergency power to a load when the input power source or mains power fails. A UPS system performs three primary functions: conditions the incoming dirty power from the utility company to give you clean, uninterruptible power, provides ride-through power to cover for sags or short-term outages, and enables seamless system shutdown during a complete power outage.

Source:
https://www.eaton.com/us/en-us/products/backup-power-ups-surge-it-power-distribution/backup-power-ups/uninterruptible-power-supply-faq.html

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:ups

Last update: **2020/11/14 14:05**

Unique Identifier (UID)

Return to Glossary

Unique Identifier is an identifier that marks that particular record as unique from every other record. It allows the record to be referenced in the Summon Index without confusion or unintentional overwriting from other records.

Source: Unique Identifier

Unit Testing

Return to Glossary

Unit Testing ensures that each part of the code developed in a component delivers the desired output. In unit testing, developers only look at the interface and the specification for a component. It provides documentation of code development as each unit of the code is thoroughly tested standalone before progressing to another unit.

Unit tests support functional tests by exercising the code that is most likely to break.

Source: https://www.simform.com/functional-testing-types/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:unittesting

Last update: **2020/11/15 02:04**

Universally Unique IDentifier (UUID)

Return to Glossary

A **Universally Unique IDentifier (UUID)** is a 128-bit number used to identify information in computer systems. The term globally unique identifier (GUID) is also used, typically in software created by Microsoft. When generated according to the standard methods, UUIDs are for practical purposes unique.

Source: https://www.google.com/search?client=firefox-b-1-d&q=UUID

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:uuid

Last update: **2020/11/15 02:09**

Universal Serial Bus (USB)

Return to Glossary

A **Universal Serial Bus (USB)** is a common interface that enables communication between devices and a host controller such as a personal computer (PC) or smartphone. It connects peripheral devices such as digital cameras, mice, keyboards, printers, scanners, media devices, external hard drives and flash drives. Because of its wide variety of uses, including support for electrical power, the USB has replaced a wide range of interfaces like the parallel and serial port.

Source: https://www.techopedia.com/definition/2320/universal-serial-bus-usb

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:usb

Last update: **2021/06/15 15:39**

UNIX

Return to Glossary

The UNIX brand has traditionally been applied to the family of multitasking, multiuser computer operating systems that derive from the original AT&T UNIX operating system, developed in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others [41].

Today, the definition of UNIX ®[42] takes the form of the worldwide Single UNIX Specification integrating X/Open Company's XPG4, IEEE's POSIX Standards and ISO C. Through continual evolution, the Single UNIX Specification is the defacto and dejure standard definition for the UNIX system application programming interfaces. As the owner of the UNIX trademark, The Open Group has separated the UNIX trademark from any actual code stream itself, thus allowing multiple implementations. Since the introduction of the Single UNIX Specification, there has been a single, open, consensus specification that defines the requirements for a conformant UNIX system.

There is also a mark, or brand, that is used to identify those products that have been certified as conforming to the Single UNIX Specification, initially UNIX 93, followed subsequently by UNIX 95, UNIX 98, UNIX 03 and now UNIX V7.

[41] “The invention of Unix”, Nokia Bell Labs, https://www.bell-labs.com/var/articles/invention-unix/

[42] “What is Unix”, The Open Grup, http://www.unix.org/what_is_unix.html

Upload Speed

Return to Glossary

Upload Speed refers to how many megabits of data per second you can send information from your computer to another device or server on the internet. While downloading information is more common, some online activities need data to travel in the opposite direction. Sending emails, playing live tournament-style video games and video calling a friend require fast upload speeds for you to send data to someone else's server.

Source: https://www.allconnect.com/blog/difference-between-download-upload-internet-speeds

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:uploadspeed

Last update: **2020/11/16 21:04**

Usability

Return to Glossary

Usability is degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. This characteristic is composed of the following sub-characteristics:

- **Appropriateness Recognizability** - Degree to which users can recognize whether a product or system is appropriate for their needs.
- **Learnability** - Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.
- **Operability** - Degree to which a product or system has attributes that make it easy to operate and control.
- **User Error Protection** - Degree to which a system protects users against making errors.
- **User Interface Aesthetics** - Degree to which a user interface enables pleasing and satisfying interaction for the user.
- **Accessibility** - Degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

Effectiveness

Return to Top

Efficiency

Return to Top

Satisfaction

Return to Top

Use Case

Return to Glossary

What is a use case?

> *A use case is a description of how a system responds to a request from an external source. This is written with reference to the technical steps that happen for the task to be completed.*

Why are use cases important?

> *Use cases are important, because they will show how systems respond when they are used. A use case describes how a system helps a user achieve their goal. Unlike a user scenario, a use case is more orientated towards the system's behavior rather than the user. The language in which a use case is written should be simple and the writer of a use case should avoid technical terminology. A use case should not include too much context about the user or their emotional response to an interaction.*

The purpose of use cases

: *In a use case a list of goals can be defined and an analysis can be made of how complex and expensive it is for these goals to be met. In team projects, use cases are useful to see which areas of a system need to be developed and what obstacles could arise in performing a task.*

Source: Use Case

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:use_case

Last update: **2021/06/09 18:31**

User Datagram Protocol (UDP)

Return to Glossary

User Datagram Protocol (UDP) is part of the Internet Protocol suite used by programs running on different computers on a network. UDP is used to send short messages called datagrams but overall, it is an unreliable, connectionless protocol.

Source: User Datagram Protocol (UDP)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:udp

Last update: **2020/11/13 02:58**

User Defined Integrity

Return to Glossary

User-Defined Integrity involves the rules and constraints created by the user to fit their particular needs. Sometimes entity, referential, and domain integrity aren't enough to safeguard data. Often, specific business rules must be taken into account and incorporated into Data Integrity measures.

Source: User-Defined Integrity

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:udefintegrity

Last update: **2021/06/09 18:23**

User Error Protection

Return to Glossary

User Error Protection is the degree to which a system protects users against making errors.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:uerror_protection

Last update: **2021/06/15 15:47**

User Interface Aesthetics

Return to Glossary

User Interface Aesthetics is the degree to which a user interface enables pleasing and satisfying interaction for the user.

Source: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/61-usability

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:interface_aesthetics

Last update: **2021/06/15 15:44**

User Scenario

Return to Glossary

User_scenario is … Source: URI

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:u:uscenario

Last update: **2021/06/15 15:56**

V

Return to Glossary

Create a Glossary entry starting with 'V' **Word or Expression →**		Add page

- Validation
- Value Chain
- Vendor Lock-In
- Vertical Scaling
- Video Privacy Protection Act (VPPA)
- Virtual Disk Image (VDI)
- Virtual Machine (VM)
- Virtual Machine Images

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:v:start

Last update: **2021/06/14 21:39**

Validation

Return to Glossary

Validation is the confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled.[43] In other words, validation ensures that “you built the right thing.”

Source: https://snebulos.mit.edu/projects/reference/NASA-Generic/NASA-STD-8739-8.pdf

[43] ISO/IEC 12207, Software life cycle processes, ISO, November 2017, Accessed 9 August 2020, https://www.iso.org/standard/63712.html

Value Chain

Return to Glossary

Value Chain is a set of activities that a firm operating in a specific industry performs in order to deliver a valuable product (i.e., good and/or service) for the market. Although it is similar to a Supply Chain, it is usually a subset of it that covers just the portion of the Supply Chain that is under the control of a single corporation or entity.

Source: https://en.wikipedia.org/wiki/Value_chain

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:v:valuechain

Last update: **2021/06/14 21:43**

Vendor Lock-In

Return to Glossary

Vendor Lock-In is the restricted or proprietary use of a technology, solution or service developed by a vendor or vendor partner. This technique can be disabling and demoralizing because customers are effectively prevented from switching to alternate vendors.

Vendor lock in is also known as proprietary lock-in or customer lock-in.

Source: Vendor Lock-In

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:v:vendorlockin

Last update: **2020/11/13 03:00**

Vertical Scaling

Return to Glossary

Vertical Scaling refers to adding more CPU, memory, or I/O resources to an existing server, or replacing one server with a more powerful server. Amazon Web Services (AWS) vertical scaling and Microsoft Azure vertical scaling can be accomplished by changing instance sizes, or in a data center by purchasing a new, more powerful appliance and discarding the old one. AWS and Azure cloud services have many different instance sizes, so scaling vertically is possible for everything from EC2 instances to RDS databases.

Source:
https://cloudcheckr.com/cloud-cost-management/cloud-vs-data-center-what-is-scalability-in-cloud-computing/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:v:verticalscaling

Last update: **2020/11/16 21:22**

Video Privacy Protection Act (VPPA)

Return to Glossary

The **Video Privacy Protection Act (VPPA) (18 U.S. Code § 2710 et seq.)** restricts the disclosure of rental or sale records of videos or similar audio-visual materials, including online streaming. Source: https://iclg.com/practice-areas/data-protection-laws-and-regulations/usa

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:v:vppa

Last update: **2021/06/15 15:51**

Virtual Disk Image (VDI)

Return to Glossary

Virtual Disk Image (VDI) is the image of a virtual hard disk or the logical disk associated with a virtual machine.

It is used in virtualization environments to create a replica of the disk space/drive assigned to one or more Virtual Machines.

Source: https://www.techopedia.com/definition/10933/virtual-disk-image-vdi

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:v:virt_disk_image

Last update: **2021/06/14 21:43**

Virtual Machine (VM)

Return to Glossary

Virtual Machine (VM) is a software program or operating system that not only exhibits the behavior of a separate computer, but is also capable of performing tasks such as running applications and programs like a separate computer.

In other words, a VM is a software application that performs most functions of a physical computer, actually behaving as a separate computer system.

A virtual machine, usually known as a guest, is created within another computing environment referred as a "host." Multiple virtual machines can exist within a single host at one time.

Source: Virtual Machine (VM)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:v:vm

Last update: **2020/11/13 15:29**

Virtual Machine Images

Return to Glossary

irtual Machine Images is a template for creating new instances. You can choose images from a catalog to create images or save your own images from running instances. Specialists in those platforms often create catalog images, making sure that they are created with the proper patches and that any software is installed and configured with good default settings. The images can be plain operating systems or can have software installed on them, such as databases, application servers, or other applications. Images usually remove some data related to runtime operations, such as swap data and configuration files with embedded IP addresses or host names.

Source: https://www.informit.com/articles/article.aspx?p=1927741&seqNum=7

W

Return to Glossary

Create a Glossary entry starting with 'W' **Word or Expression** →	□ Add page

- WaitSet
- Waterfall Model
- Web Application (Web App)
- Weight of Network
- White Box Testing
- Wide Area Network (WAN)
- Will (Requirement)
- Windows Registry
- Wire Protocol
- Wired Network
- Wireless Fidelity (Wi-Fi)
- Wireless Network
- Wizard

=-

WaitSet

Return to Glossary

A **WaitSet** allows application threads to block and wait until one or more of the attached Conditions has a trigger value of TRUE, or until a specified timeout occurs.

Source: OpenSplice Glossary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:waitset

Last update: **2021/06/14 21:42**

Waterfall Model

Return to Glossary

The **Waterfall Model** is a sequential software development process model that follows the following defined phases:

1. Conception
2. Initiation
3. Analysis
4. Design
5. Construction
6. Testing
7. Production/Implementation
8. Maintenance

Using the software development life cycle's (SDLC) common steps, the waterfall model enforces moving to the next phase only after completion of the previous phase. Returning to a previous phase is frowned upon unless there is a clear need to do so.

Source: Waterfall Model

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:waterfall

Last update: **2020/11/14 20:03**

Web Application (Web App)

Return to Glossary

A **Web Application (Web App)** is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system (OS) of the device. Web applications are accessed by the user through a web browser with an active network connection. These applications are programmed using a Client-Server modeled structure—the user (“Client”) is provided services through an off-site Server that is hosted by a third-party. Examples of commonly-used web applications include: web-mail, online retail sales, online banking, and online auctions.

Source: https://en.wikipedia.org/wiki/Web_application

Weight of Network

Return to Glossary

Weight of Network is a figure given to all the coins that are actively Staking on the entire XLR Network. The Coins in your Wallet that are available for Staking also has a Weight. If the Weight of the Network is 8000 and the Weight of your Coins is 2000 then you have a good chance of Minting some new coins. However if the Weight of your Coins is just 1 then the chance of you Minting some new coins are remote. Some Networks combine other factors to the calculation of weight such as age of the coins (i.e., older coins are heavier).

Source: Masternode vs Staking

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:weight_of_network

Last update: **2020/11/13 02:57**

White Box Testing

Return to Glossary

White Box Testing techniques analyze the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing.

Source: https://www.geeksforgeeks.org/software-engineering-white-box-testing/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:whiteboxtesting

Last update: **2020/12/20 21:02**

Wide Area Network (WAN)

Return to Glossary

A **Wide Area Network (WAN)** is a network that exists over a large-scale geographical area. A WAN connects different smaller networks, including Local Area Network (LAN) and metro area networks (MANs). This ensures that computers and users in one location can communicate with computers and users in other locations. WAN implementation can be done either with the help of the public transmission system or a private network.

Source: Wide Area Network (WAN)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:wan

Last update: **2020/11/13 02:57**

Will (Requirement)

Return to Glossary

Will is not per se the specification of a requirement but is used to indicate a statement of fact. Will statements are not subject to verification. For example: if I want to tell you something about another system I will use "will". "The XYZ system will have the timing as defined in an ICD 1234." Or if I am giving a description of something that exists today, I will use "will". "This report will contain this data…" In a statement of work (SOW) or task order for a vendor or supplier, I use will to communicate something I will do for or provide to the vendor or supplier. .

Source: https://reqexperts.com/2012/10/09/using-the-correct-terms-shall-will-should/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:will_req

Last update: **2021/06/14 21:10**

Windows Registry

Return to Glossary

Windows Registry is a hierarchically structured database that is used to store data related to configuration settings, software and user preferences in a Microsoft Windows Operating System (OS). It contains entries and values that control the behavior of certain configurations and user preferences, as well as information for OS components and applications that operate at a low level.

Most Windows applications write entries into the Windows registry during the installation process.

Source: https://www.techopedia.com/definition/5001/windows-registry

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:windowsregistry

Last update: **2020/11/13 02:57**

Wired Network

Return to Glossary

A **Wired Network** uses cables to connect devices, such as laptop or desktop computers, to the Internet or another network. A wired network has some disadvantages when compared to a wireless network. The biggest disadvantage is that your device is tethered to a router. The most common wired networks use cables connected at one end to an Ethernet port on the network router and at the other end to a computer or other device.

Source: Wired Network

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:wired

Last update: **2020/11/14 14:09**

Wireless Fidelity (Wi-Fi)

Return to Glossary

Wireless Fidelity (Wi-Fi) is a type of wireless network technology used for connecting to the Internet. The frequencies wi-fi works at are 2.4Ghz or 5Ghz, ensure no interference with cellphones, broadcast radio, TV antenna and two-way radios are encountered during transmission.

To simplify, Wi-Fi is basically just radio waves broadcast from a Wi-Fi router, a device detecting and deciphering the waves, and then sending back data to the router. It works very similarly to an AM/ FM radio but it is two-way communication channel. Wi-Fi works over longer distances than Bluetooth or infrared and is also a low power unobtrusive technology, making it suitable for portable devices such as laptops and palmtops. Wi-Fi is governed by the Wi-Fi Alliance, an association of manufacturers and regulators defining standards and certifying products as Wi-Fi compatible.

Source: Wireless Fidelity (Wi-Fi)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:wifi

Last update: **2020/11/14 17:39**

Wireless Network

Return to Glossary

A **Wireless Network** is computer network not connected by cables of any kind. The use of a wireless network enables enterprises to avoid the costly process of introducing cables into buildings or as a connection between different equipment locations. The basis of wireless systems are radio waves, an implementation that takes place at the physical level of network structure.

Source: Wireless Network

Wire Protocol

Return to Glossary

Wire Protocol refers to a way of getting data from point-to-point: A Wire Protocol is needed if more than one application has to interoperate. It generally refers to protocols higher than the physical layer. In contrast to transport protocols at the transport level (like Transmission Control Protocol (TCP) or User Datagram Protocol (UDP)), the term “wire protocol” is used to describe a common way to represent information at the Application Level of the Open Systems Interconnection (OSI) Model. It refers only to a common Application Layer protocol and not to a common object semantic[clarification needed] of the applications. Such a representation at application level needs a common XML Information Set (XML Infoset) and a data binding (using e.g. a common encoding scheme like XML Schema Definition (XSD)).

It generally refers to higher layers, including Ethernet and ATM (layer 2) and even higher layer distributed object protocols such as SOAP, CORBA or RMI.

The Wire Protocol may be either text-based or a binary protocol. Although an important architectural decision, this is a separate matter from the distinction between Wire Protocols and programmatic Application Programming Interfaces (APIs).

Source: Wire Protocol

Wizard

Return to Glossary

A **Wizard** or **Setup Assistant** is a user interface type that presents a user with a sequence of dialog boxes that lead the user through a series of well-defined steps. Tasks that are complex, infrequently performed, or unfamiliar may be easier to perform using a wizard.

Source: https://en.wikipedia.org/wiki/Wizard_(software)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:w:wizard

Last update: **2020/11/13 02:57**

X

Return to Glossary

Create a Glossary entry starting with 'X' **Word or Expression →**	□	Add page

- eXtensible Markup Language (XML)
- XML Information Set (XML Infoset)
- XML Schema Definition (XSD)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:x:start

Last update: **2021/06/14 21:40**

eXtensible Markup Language (XML)

Return to Glossary

eXtensible Markup Language (XML) is a universal format, maintained by the W3C, used for representation and transfer of structured data on the web or between different applications.

The language uses a structured representation by allowing users to create custom defined tags according to XML Document Type Definition (DTD) standards. The structure of an XML document can be represented in the form of a tree known as a Document Object Model (DOM).

Source: eXtensible Markup Language (XML)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:x:xml

Last update: **2020/11/13 15:29**

XML Information Set (XML Infoset)

Return to Glossary

XML Information Set (XML Infoset) is a W3C specification describing an abstract Data Model (DM) of an XML document in terms of a set of information items.[1] The definitions in the XML Information Set specification are meant to be used in other specifications that need to refer to the information in a well-formed XML document.

An XML document has an information set if it is well-formed and satisfies the namespace constraints. There is no requirement for an XML document to be valid in order to have an information set.

Source: XML Information Set (XML Infoset)

XML Schema Definition (XSD)

Return to Glossary

XML Schema Definition (XSD), a recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. It can be used by programmers to verify each piece of item content in a document. They can check if it adheres to the description of the element it is placed in.

Like all eXtensible Markup Language (XML) schema languages, XSD can be used to express a set of rules to which an XML document must conform in order to be considered “valid” according to that schema. However, unlike most other schema languages, XSD was also designed with the intent that determination of a document's validity would produce a collection of information adhering to specific data types. Such a post-validation XML Information Set (XML Infoset) can be useful in the development of XML document processing software.

Source: XML Schema Definition(XSD)

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:x:xsd

Last update: **2020/11/13 15:29**

Y

Return to Glossary

Create a Glossary entry starting with 'Y' **Word or Expression →**	Add page

No subnamespaces.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:y:start

Last update: **2021/06/14 21:40**

Z

Return to Glossary

Create a Glossary entry starting with 'Z' **Word or Expression →**	▭ Add page

- ZigBee

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.a_glossary:z:start

Last update: **2021/06/14 21:41**

ZigBee

Return to Glossary

ZigBee is an open global standard for wireless technology designed to use low-power digital radio signals for personal area networks. ZigBee operates on the IEEE 802.15.4 specification and is used to create networks that require a low data transfer rate, energy efficiency and secure networking. It is employed in a number of applications such as building automation systems, heating and cooling control and in medical devices.

ZigBee is designed to be simpler and less expensive than other personal are network technologies such as Bluetooth.

Source: ZigBee

Appendix B: Standards Organizations

Return to Reference Architecture (RA) or Return to Appendices

The DIDO RA recognizes two categories of standard bodies:

- Technical Standards Bodies
- de facto Standard Bodies

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.b_stds

Last update: **2021/06/13 15:19**

Technical Standards Bodies

return to Standards Area

Technical standards are developed by Standards Organizations [8]. Standards organizations are organizations whose primary activities are to develop, coordinate, promulgate, revise, amend, reissue, interpret, or otherwise produce technical standards intended to address the needs of a group of affected adopters.[9]

The DIDO RA provides data sheets for technical standards developed by the following organizations:

- Apache Software Foundation (ASF)
- ECMA International
- Institute of Electrical and Electronics Engineers (IEEE)
- Internet Engineering Task Force (IETF)
- International Organization for Standardization (ISO)
- International Telecommunications Union (ITU)
- National Institute of Standards and Technology (NIST)
- Organization for the Advancement of Structured Information Standards (OASIS)
- Object Management Group (OMG)
- Open Source Initiative (OSI)
- World Wide Web Consortium (W3C)

[8] also known as standards body, Standards Developing Organization (SDO), or Standards Setting Organization (SSO)

[9] Intercloud: Solving Interoperability and Communication in a Cloud of Clouds, Appendix A, by Ken Owens, Monique Morrow, Venkata Josyula, Jazib Frahim, Publisher: Cisco Press Release Date: June 2016 ISBN: 9780134189239,https://learning.oreilly.com/library/view/intercloud-solving-interoperability/9780134189239/app01.html

de facto Standards Bodies

return to Standards Area

A *de facto* Standard is something that is used so widely that it is considered a standard for a given application although it has no official status. [10] A *de facto* Standard is generally controlled by a corporation (Microsoft, IBM, Google, Amazon, etc) or group (Apache, Linux, Mozilla, etc).

The DIDO RA provides data sheets for de facto standards developed by the following:

Corporate Projects

- Amazon
- Apache Software Foundation (ASF)
- Apple
- Bitcoin
- Consortium for Information & Software Quality (CISQ)
- Ethereum
- Google
- IOTA
- Linux Foundation
- Microsoft
- Oracle
- Talk Openly Develop Openly (TODO)

Individual Projects

- GIT (Revision Control)
- InterPlanetary File System (IPFS)
- Jenkins (Continuous Delivery)
- Jira (Bug tracking system)
- Participating in Open Source Communities
- ZeroMQ Distributed Messaging
- ZeroMQ Message Transport Protocol (ZMTP)

[10] https://whatis.techtarget.com/definition/de-facto-standard

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.b_stds:defact

Last update: **2021/06/18 13:39**

Appendix C: Hardware Architectures

Return to Reference Architecture (RA) or Return to Appendices or 4.2.1.1 Hardware Platform

Gezelter[11] defined Hardware architecture as follows:

> *"Computer architecture" refers to the underlying physical and logical structures of the computer system. In the case of a computer system, this includes instruction set, numeric sizes and representations, and how the system connects to external devices (interrupts or polling).*
>
> *While in the theoretical sense, most architectures are functionally equivalent, it does not mean that some architectural choices make some applications and algorithms easier or more efficient to implement. An architecture without support for floating point can be extended in software to perform the needed functions, but it will generally be slower than a hardware implementation. In the interfacing device area, interrupts make it more efficient to overlap device operation with computing. Such operations can be done without interrupts, but it is far more complex.*
>
> *In summary, computer architecture defines the underlying structure of the computing system. It governs how the various elements interact.*

The following Hardware Architectures need to be considered when designing distributed systems. It is not important that every distributed system supports all these kinds of architectures, but that the inclusion and elimination of some of the architectures is an overt act. The Goal during the functional requirements specification process is to establish the subset of Hardware Architectures supported. This does not mean that the set is static and, thus, cannot be changed. It does mean; however, that changes require careful consideration and planning.

- C.1 Embedded Systems
- C.2 Servers
- C.3 Desktops
- C.4 Handheld Computers
- C.5 Supercomputers
- C.6 Network Devices

[11] Gzelter, Robert; The Graduate Center, CUNY, Quora, 2018, Accesssed 8 December 2020, https://www.quora.com/What-is-the-importance-of-computer-architecture-in-a-computer-system

C.1 Embedded Systems

Return to Hardware Architectures

An **Embedded System** is a computer custom built to serve a specific purpose. Some examples of special purpose computers are cash registers, calculators, smart thermostats, and engine monitors in vehicles. A general purpose computer is one designed to perform general tasks: e.g., tablets, laptops, desktops, workstations and servers. Recently, smart phones have crossed over from being special purpose computers to general purpose computers, performing lots of tasks such as browsing the internet, playing songs, watching films, gaming and taking photos. They are no longer limited to making and receiving phone calls.

Although the general purpose computers are almost ubiquitous and can solve many problems associated with embedded computer, the need for specialized microprocessors has increased dramatically. General purpose computers are not necessarily best when it comes to specialized needs. They are often cumbersome, expensive and not well suited to handle specific needs. For example, a smart thermostat, a calculator or security sensors are best left to specialized computers.

Embedded systems can be classified into three categories:

- C.1.1 Embedded Subsystem
- C.1.2 Standalone Embedded Systems
- C.1.3 Networked Embedded Systems

Often these systems use a Microcontroller or microprocessor.

A microcontroller is a chip optimized to control electronic devices. It is stored in a single integrated circuit which is dedicated to performing a particular task and execute one specific application. It is specially designed circuits for embedded applications and is widely used in automatically controlled electronic devices. It contains memory, processor, and programmable I/O.

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.c_hwarch:1_embedded

Last update: **2021/06/14 12:05**

C.1.1 Embedded Subsystem

Return to Embedded Systems

About

An **Embedded Subsystems** is part of a larger system. For example, the temperature sensor in refrigerators and freezers, the rain sensor in windshield wipers or a motion detector in a light are examples of an embedded system. As a standalone independent system, these subsystems are of little value. They only become valuable when they are incorporated into a large system.

DIDO Specifics

To be added/expanded in future revisions of the DIDO RA

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.c_hwarch:1_embedded:subsystems

Last update: **2021/06/14 12:07**

C.1.2 Standalone Embedded Systems

Return to Embedded Systems

About

Standalone Embedded Systems are devices that can perform tasks on their own, but can be also used as part of a larger system. Some examples are USB Devices which can store data and can exist independently of other systems, but can also be part of a larger system to perform operations like data storage. Other examples might be digital watches or bathroom scales.

DIDO Specifics

To be added/expanded in future revisions of the DIDO RA

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.c_hwarch:1_embedded:standalone

Last update: **2021/06/14 12:08**

C.1.3 Networked Embedded Systems

Return to Embedded Systems

About

Networked Embedded Systems - are devices that are networked together as peers to provide a “system”. Some examples would be the sensors in a home security or home automation system. Some networked systems can contain hundreds of thousands of peers all working in harmony to monitor large operations such as a dam, an air traffic control system, or a hospital.

DIDO Specifics

To be added/expanded in future revisions of the DIDO RA

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.c_hwarch:1_embedded:networked

Last update: **2021/06/14 12:18**

C.2 Servers

Return to Hardware Architectures

About

The term **Server** is an overloaded term which can sometimes lead to confusion. All the uses of the term Server imply a Client/Server model wherein a Client makes a request to a Server that fulfills that request and provides a response back to the Client. The Client/Server model can be chained together so that a Server can become Client to other Servers.

A Server is either a computer program or device providing services to other computer programs or devices. The users of the service is referred to as a client. Some examples of software servers are mail servers, database servers, web servers, application servers, etc. The physical device (i.e., computer) that hosts the software servers is also referred to as a server. The physical server can be dedicated to host a single software service (i.e., mail server) or it can be used to host multiple services. Complicating the issue is that sometimes a single software service might span across multiple physical servers.

- C.2.1 Software Servers
- C.2.2 Hardware Servers

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.c_hwarch:server

Last update: **2021/06/14 12:19**

C.2.1 Software Servers

Return to Servers

About

Minimally, a Software Server requires two software components: an operating platform and an application. The operating platform provides the runtime infrastructure needed to support an application. The platform can include the hardware (real or virtual), the network, the operating system (OS), the dependent services and the software libraries needed to access the services, operating system network and hardware.[33)]

In order to communicate with the clients in the outside world, typical Software Servers generally use domain names, TCP IP Addresses, ports and MAC addresses. These can be statically assigned or dynamic, as well as, virtual and real.

Some common Software Servers are:

- **Web Servers** are computer programs that serve requested HTML pages or files. In this case, a web browser acts as the client.
- **Application Servers** are programs running on a distributed network that provide the business logic for an application program.
- **Proxy Servers** acts as the intermediary between endpoint devices, such as a computer, and another server from which a user or client is requesting a service.
- **Mail Servers** receive incoming emails from local users (i.e., individuals or applications) within the same domain and from remote senders and forwards outgoing emails for delivery.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[33)] Brian M. Psey, Server, Whatis, Accessed 8 December 2020, https://whatis.techtarget.com/definition/server

C.2.2 Hardware Servers

Return to Servers

About

Hardware Servers can vary widely: from personal computers (i.e., Desktop Computers) to specially built rack mounted computers depending on the amount of work that needs to be done and the number of clients that need to be supported. For example, quite often database servers are software that can run on a personal computer to service the needs of an individual using that machine or can be on many rack mounted servers supporting thousands to millions of clients. As a general rule, Servers are more powerful and expensive than the client's hardware connected to them. The power of a server is not just measured by the speed of its CPU, but also in its ability to handle multiple events and connections and its memory access speed.

Many servers, whether running locally on a personal computer or remotely on large clusters of servers, are accessed over a network – usually the Ethernet. They are intended to run unattended without Human-machine interface (HMI) or Graphical User Interface (GUI). These servers are configured and managed remotely using software specifically designed for that purpose. They are typically managed using web based interfaces accessible via browsers over secure connections and POSIX compliant interfaces.

Historically, Servers used Central Processing Unit (CPU) that were either Complex Instruction Set Computer (CISC) or Reduced Instruction Set Computer (RISC) architectures. While the Central Processing Units (CPUs) are designed to quickly handle a wide-range of general tasks sequentially, they are not well suited to handling many small tasks that in parallel. Recently some servers have been built using General-Purpose Graphics Processing Units (GPGPUs) instead of CPUs. Despite their name (GPGPU), these Servers are not used to process lots of graphics or images, instead, they are used to perform lots of small tasks in parallel (i.e., concurrently).

Consequently, GPGPU based servers performing parallel operations on multiple sets of data are being employed to support non-graphical tasks such as machine learning and scientific computation. Designed with thousands of processor cores running simultaneously, GPGPUs enable massive parallelism with each core performing efficient calculations simultaneously.

Some common Software Servers are:

- **File Servers** are responsible for the central storage and management of data files
- **Policy Servers** are security components of a policy-based network that provides authorization services and facilitates tracking and control of files
- **Print Server** provides access to one or more network-attached printers
- **Media Server** provides access to streaming video and audio
- **Game Server** provides access to the authoritative source of events shared across users in multiplayer video games
- **Domain Servers** provides Internet Domain Name translation to internet addresses

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

C.3 Desktops

Return to Hardware Architectures

About

Desktop computers (sometimes referred to as personal computers) are designed to meet regular general purpose computational needs at a single location such as on or near a desk or table. Desktop computers, unlike handheld computers, generally require more power to support peripherals such as disks and monitors. Consequently, even though Desktop computers can be moved their power requirements make them cumbersome. A desktop's peripherals include: one or more monitors, internal and external disks, mice, printers, and oftentimes Wired Network Ethernet connections. A stereotypical configuration generally comprises a case housing the power supply, a Motherboard with a Central Processing Unit (CPU); a keyboard and mouse for input; a computer monitor; speakers; and usually a printer for output. The case may be horizontally or vertically oriented, placed underneath, beside, or on top of a desk.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.c_hwarch:desktop

Last update: **2021/06/14 13:33**

C.4 Handheld Computers

Return to Hardware Architectures

About

Handheld computers (or mobile devices) are computers small enough to hold in one's hand or intended to be move easily from place to place. Most modern handheld computers use an integrated touchscreen for controlling most user interfaces with a minimal Human-machine interface (HMI). Most handheld computers connect to the Ethernet using Wireless Network connections such as Wireless Fidelity (Wi-Fi), Bluetooth, ZigBee, and/or Near-Field-Communication (NFC).

Handheld devices provide additional features such as integrated cameras, the ability to support voice and video calls, video games, and the Global Positioning System (GPS). These additional features represent enhancements in security such as face recognition, fingerprint scanners, geographic locations, barcode readers, Radio Frequency Identification (RFID) readers, smart card readers, accelerometer, magnetometers and gyroscopes. Power is usually provided using lightweight portable batteries.

Usually, these handheld devices (i.e., phones, tablets, etc.) use an Operating System (OS) that is specifically designed to support portability, such as Android or iOS, and that are extensible using third party software offerings available through an OS specific "store".

However, some of these "handheld" devices have evolved into non-handheld usages, e.g., smart watches, automotive displays, smart refrigerators, and cameras.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

C.5 Supercomputers

Return to Hardware Architectures

About

A **Supercomputer**[45] is a computer performing at or near the current highest operational rate for computers. Traditionally, supercomputers have been used for scientific and engineering applications: to handle very large databases, or to do a great amount of computation, or both. Although advances such as Multi-core Processor and General-Purpose Graphics Processing Unit (GPGPU) have enabled powerful machines for personal use (see: desktop supercomputer, GPU supercomputer), by definition, a supercomputer is exceptional in terms of performance.

According to Rouse[46], at any given time, there are a few well-publicized supercomputers that operate at extremely high speeds relative to all other computers. The term is also sometimes applied to far slower (but still impressively fast) computers. The largest, most powerful supercomputers are really multiple computers that perform Parallel Processing. In general, there are two parallel processing approaches: Symmetric Multiprocessing (SMP) and Massively Parallel Processing (MPP).

Rouse[47] reports that as of June 2016, the fastest supercomputer in the world was the Sunway TaihuLight, in the city of Wixu in China. A few statistics on TaihuLight:

- 40,960 64-bit, RISC processors with 260 cores each.
- Peak performance of 125 petaflops (quadrillion floating point operations per second).
- 32GB DDR3 memory per compute node, 1.3 PB memory in total.
- Linux-based Sunway Raise operating system (OS).

Some of the uses for supercomputers are[48]:

- Recreating the Big Bang
- Understanding Earthquakes
- Folding proteins
- Mapping the blood stream
- Modeling swine flu
- Testing nuclear weapons
- Forecasting hurricanes
- Building Brains

More recently, supercomputers have been used to model how the Coronavirus spreads.[49]

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[45], [46], [47] Rouse, Margret; Definition of supercomputer, WhatIs.com, Accessed 8 December 2020, https://whatis.techtarget.com/definition/supercomputer

[48] Pappas, Stephanie; 9 Super-Cool Uses for Supercomputers, livescience.com, 30 April 2010, Accessed 8 November 2020, https://www.livescience.com/6392-9-super-cool-supercomputers.html

[49] Kelleher, Suzanne Rowan; Japanese Supercomputer Shows How Coronavirus Spreads In A Dining Setting, Forbes, 19 October 2020, Accessed 8 December 2020, https://www.forbes.com/sites/suzannerowankelleher/2020/10/19/viral-video-japanese-supercomputer-shows-how-coronavirus-spreads-in-a-dining-setting/?sh=57971027333f

C.6 Network Devices

Return to Hardware Architectures

About

Networking Devices, also known as **Network Equipment** or **Computer Networking Devices**, are electronic devices required for communication and interaction between devices on a computer network. In essence, these devices mediate data transmission in networks. An Endpoint is an Network Node and is the last receptor or generator of the network traffic.

Figure 9 is an example of a network that represents many of the components of a network. Many of these devices have their own operating systems tailored specifically to optimize the flow and control of network traffic. In addition to the devices shown in the diagram, there are other network devices such as Network Attached Storage (NAS).

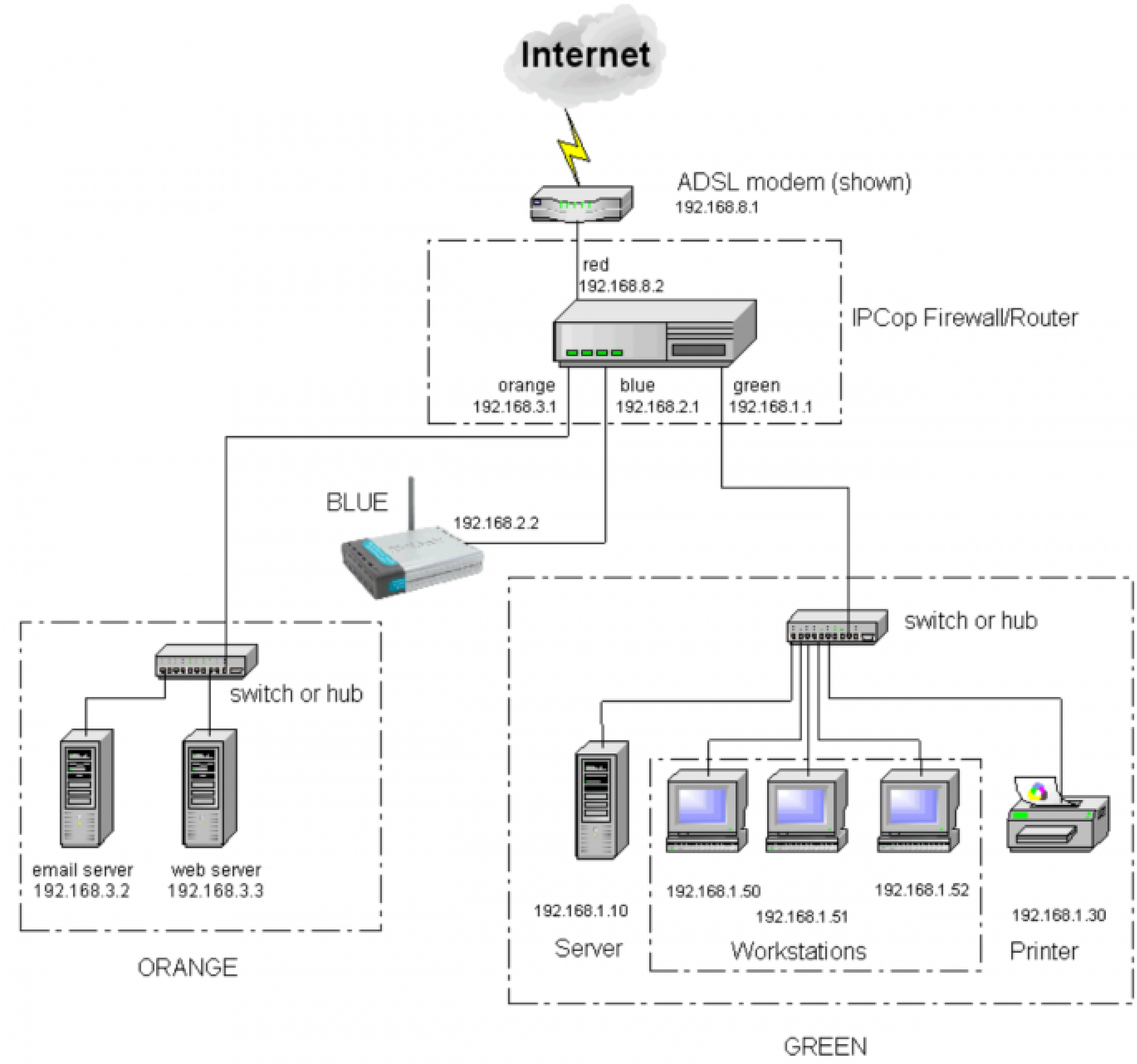


Figure 9: Example of a network Architecture[50]

Table ##REF:netDevSumary## Provides a summary of the device found in the example network depicted in Figure 9.

<table netDevSumary> <caption>A summary of common devices found in networks.</caption>

Network Component	**Description** [51]
Hub	A hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, such as, for example, the connector in a star topology that connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices.

Network Component	Description [51]
Switch	A switch is a multiport bridge with a buffer and a design that can boost its efficiency (a large number of ports imply less traffic) and performance. A switch is a data link layer device.
Router	A router is a device like a switch that routes data packets based on their IP addresses. A Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and use a dynamically updated routing table to make decisions regarding data packet routing. Routers divide broadcast domains of hosts connected through it.
Bridge	A bridge operates at the data link layer. A bridge is a repeater, with the ability to filter content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs using the same protocol. It has a single input and single output port, thus making it a 2 port device.
Gateway	A gateway, as the name suggests, is a passage to connect two networks together that may work in accordance with different networking models. A gateway basically functions as a messenger agent that takes data from one system, interprets it, and transfers it to another system. Gateways are also called protocol converters and can operate at any network layer. Gateways are generally more complex than a switch or router.
Modem	Transmission mode refers to the manner by which data is transferred between between two devices. It is also known as communication mode. Buses and networks are designed to allow communication to occur between individual devices that are interconnected. There are three types of transmission modes: • Simplex Mode • Half-Duplex Mode • Full-Duplex Mode
Repeater	A repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the distance over which the signal can be transmitted over the same network.
Network Appliance	A type of computing appliance that aids in the flow of information to other network-connected computing devices. Services that may be provided by a network appliance include firewall functions, caching, authentication, network address translation and IP address management. [52]

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

[50] Vorgu Topology, Accessed 10 December 2020, https://wiki.itcollege.ee/index.php?title=File:Vorgu_topoloogia.png&limit=500

[51] FeekForGeeks, Network Devices (Hub, Repeater, Bridge, Switch, Router, Gateways and Brouter), 12 January 2020, Accessed: 10 December 2020; https://www.geeksforgeeks.org/network-devices-hub-repeater-bridge-switch-router-gateways/

[52]

Gartner, Network Appliance, Accessed: 10 December 2020; https://www.gartner.com/en/information-technology/glossary/network-appliance#:~:text=A%20type%20of%20computing%20appliance,translation%20and%20IP%20address%20management.

Appendix D: Operating Systems

Return to Reference Architecture (RA) or Return to Appendices

Create a Operating System datasheet for **OS Name** →		Add page

- Android
- Apstra
- Azure Real Time Operating System (or Azure RTOS)
- Azure Sphere OS
- balenaOS
- Blackberry QNX
- CentOS
- Chromium OS
- Cisco Digital Network Architecture (Cisco DNA)
- Cisco Internetwork Operating System (IOS)
- Cisco IOS XR
- Cisco NX-OS
- ClearOS
- CloudReady
- ExtremeXOS
- FreeBSD
- FreeRTOS
- IBM i
- iOS
- Junos operating system (Junos OS)
- LynxOS RTOS
- Nokia X Software Platform
- Open Network Linux
- OpenServer
- Oracle Linux (OL)
- Oracle Solaris
- Red Hat Enterprise Linux (RHEL)
- SANtricity Software Operating System (OS)
- SCO UnixWare
- SUSE Linux Enterprise Server (SLES)
- TrueNAS
- Ubuntu Linux
- Windows
- Windows IoT
- Windows NT
- Windows Server
- Windows XP

Android

Return to Operating Systems

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in November 2007, with the first commercial Android device launched in September 2008.

It is free and open source software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License. However most Android devices ship with additional proprietary software pre-installed, most notably Google Mobile Services (GMS) which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform. About 70 percent of Android smartphones run Google's ecosystem; competing Android ecosystems and forks include Fire OS (developed by Amazon) or LineageOS. However the “Android” name and logo are trademarks of Google which impose standards to restrict “uncertified” devices outside their ecosystem to use Android branding.

The source code has been used to develop variants of Android on a range of other electronics, such as game consoles, digital cameras, portable media players, PCs and others, each with a specialized user interface. Some well known derivatives include Android TV for televisions and Wear OS for wearables, both developed by Google. Software packages on Android, which use the APK format, are generally distributed through proprietary application stores like Google Play Store, Samsung Galaxy Store, and Huawei AppGallery, or open source platforms like Aptoide or F-Droid.

Source: https://en.wikipedia.org/wiki/Android_(operating_system)

Datasheet	
Developer	Various (mostly Google and the Open Handset Alliance)
Written in	Java (UI), C (core), C++ and others
OS family	Unix-like (Modified Linux kernel)
Working state	Current
Source model	Open source (most devices include proprietary components, such as Google Play)
Initial release	September 23, 2008;
Latest release	Android 11 / September 8, 2020;
Available in	https://android.googlesource.com/
Platforms	64-bit (32-bit being dropped) ARM, x86 and x86-64
Kernel type	Linux kernel
Userland	Bionic libc, mksh shell, Toybox as core utilities (beginning with Android 6.0)
Default user interface	Graphical (multi-touch)
License	Apache License 2.0 for userspace software GNU GPL v2 for the Linux kernel modifications
Official Website	https://www.android.com/

Apstra

Return to Operating Systems

EOS is **Arista's** network operating system, and comes as one image that runs across all Arista devices or in a virtual machine (VM). EOS runs on an unmodified Linux kernel under a Fedora-based userland. There are more than 100 independent regular processes, called agents, responsible for different aspects and features of the switch, including drivers that manage the switching application-specific integrated circuit (ASICs), the command-line interface (CLI), Simple Network Management Protocol (SNMP), Spanning Tree Protocol, and various routing protocols. All the state of the switch and its various protocols is centralized in another process, called Sysdb. Separating processing (carried by the agents) from the state (in Sysdb) gives EOS two important properties. The first is software fault containment, which means that if a software fault occurs, any damage is limited to one agent. The second is stateful restarts, since the state is stored in Sysdb, when an agent restarts it picks up where it left off. Since agents are independent processes, they can also be upgraded while the switch is running (a feature called ISSU – In-Service Software Upgrade).

The fact that EOS runs on Linux allows the usage of common Linux tools on the switch itself, such as tcpdump or configuration management systems. EOS provides extensive application programming interfaces (APIs) to communicate with and control all aspects of the switch. To showcase EOS' extensibility, Arista developed a module named CloudVision that extends the CLI to use Extensible Messaging and Presence Protocol (XMPP) as a shared message bus to manage and configure switches. This was implemented simply by integrating an existing open-source XMPP Python library with the CLI.

Source: https://apstra.com/products/

Datasheet	
Developer	Apstra
Written in	C, Assembly
OS family	Linux
Working state	Current
Source model	Open source and Closed source
Initial release	2014
Latest release	September 2015
Available in	English
Userland	Fedora-based
Platforms	IA-32, x86-64, Itanium
Kernel type	Monolithic
Default user interface	Command-line
License	Open source and Proprietary

Azure Real Time Operating System (or Azure RTOS)

Return to Operating Systems

Microsoft's **Azure Real Time Operating System (or Azure RTOS)** is an embedded development suite including what Microsoft describes as a small but powerful operating system that provides reliable, ultra-fast performance for resource-constrained devices. It is a small, fast, reliable, and easy-to-use RTOS for connecting deeply embedded IoT devices. They present their solution as easy-to-use and "market-proven, having been deployed on more than 10 billion devices worldwide." .

Sources: https://www.trustradius.com/operating-systems?f=0, https://www.infoworld.com/article/3536569/inside-microsofts-latest-os-azure-rtos.html

Datasheet	
Developer	Microsoft
Written in	C
OS family	Real-time operating system (RTOS)
Working state	Current
Source model	Source-available software
Initial release	February 1, 2010
Latest release	v6.1.2_rel[1] / November 10, 2020
Available in	English
Platforms	ARC, ARM, Blackfin, CEVA, C6x, MIPS, NXP, PIC, PowerPC, RISC-V, RX, SH, SHARC, TI, V850, Xtensa, x86, Coldfire, & others
Kernel type	Embedded, deterministic, real-time microkernel, picokernel
Default user interface	Embedded UI support (GUIX)
License	Proprietary

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:axure_rtos

Last update: **2021/06/09 23:09**

Azure Sphere OS

Return to Operating Systems

Azure Sphere OS is a custom Linux-based microcontroller operating system created by Microsoft to run on an Azure Sphere-certified chip and to connect to the Azure Sphere Security Service. The Azure Sphere OS provides a platform for Internet of Things application development, including both high-level applications and real-time capable applications. It is the first operating system running a Linux kernel that Microsoft has publicly released and the second Unix-like operating system that the company has developed for external (public) users, the other being Xenix.

Source: https://en.wikipedia.org/wiki/Azure_Sphere

Datasheet	
Developer	Microsoft
Written in	C and others
OS family	Unix-like (Linux)
Working state	Current
Source model	At least partially open source
Initial release	
Latest release	19.10, released 2019; 24 February 2020
Available in	English
Platforms	ARM (MediaTek MT3620)
Kernel type	Monolithic Kernel
Default user interface	
License	TBD
Official Website	https://azure.microsoft.com/en-us/services/azure-sphere/

balenaOS

Return to Operating Systems

BalenaOS is an operating system optimized for running Docker containers on embedded devices, with an emphasis on reliability over long periods of operation, as well as a productive developer workflow inspired by the lessons learned while building balena.

The core insight behind balenaOS is that Linux containers offer, for the first time, a practical path to using virtualization on embedded devices. VMs and hypervisors have lead to huge leaps in productivity and automation for cloud deployments, but their abstraction of hardware, as well as their resource overhead and lack of hardware support, means that they are not suitable for embedded scenarios. With OS-level virtualization, as implemented for Linux containers, both those objections are lifted for Linux devices, of which there are many in the Internet of Things.

BalenaOS is an operating system built for easy portability to multiple device types (via the Yocto framework and optimized for Linux containers, and Docker in particular. There are many decisions, large and small, we have made to enable that vision, which are present throughout our architecture.

The first version of balenaOS was developed as part of the balena platform, and has run on thousands of embedded devices on balena, deployed in many different contexts for several years. balenaOS v2 represents the combination of the learnings we extracted over those years, as well as our determination to make balenaOS a first-class open source project, able to run as an independent operating system, for any context where embedded devices and containers intersect.

Source: https://www.balena.io/docs/reference/OS/overview/2.x/

Datasheet	
Developer	Balena
Written in	JavaScript, HTML, GO
OS family	Yocto Linux-based
Working state	current
Source model	Open source
Initial release	
Latest release	
Available in	
Platforms	
Kernel type	
Default user interface	
License	

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:balenaos

Last update: **2020/12/08 14:38**

Blackberry QNX

Return to Operating Systems

Blackberry QNX is a microkernel-based OS, QNX is based on the idea of running most of the operating system kernel in the form of a number of small tasks, named Resource Managers. This differs from the more traditional monolithic kernel, in which the operating system kernel is one very large program composed of a huge number of parts, with special abilities. In the case of QNX, the use of a microkernel allows users (developers) to turn off any functions they do not need without having to change the OS. Instead, such services will simply not run.

To demonstrate the OS's capability and relatively small size, in the late 1990s QNX released a demo image that included the POSIX-compliant QNX 4 OS, a full graphical user interface, graphical text editor, TCP/IP networking, web browser and web server that all fit on a bootable 1.44 MB floppy disk.

Source: https://en.wikipedia.org/wiki/QNX

Datasheet	
Developer	BlackBerry
Written in	
OS family	Unix-like
Working state	Current
Source model	Closed Source
Initial release	1982
Latest release	7.1 / July 2020
Available in	
Platforms	x86, MIPS, PowerPC, SH-4, ARM, StrongARM, XScale
Kernel type	RTOS (microkernel)
Default user interface	
License	Proprietary, license for noncommercial and academic users.
www.qnx.com	www.qnx.com

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:blackberry_qnx

Last update: **2020/12/02 16:25**

CentOS

Return to Operating Systems

CentOS (/'sɛntɒs/, from Community Enterprise Operating System) is a Linux distribution that provides a free, community-supported computing platform functionally compatible with its upstream source, Red Hat Enterprise Linux (RHEL). In January 2014, CentOS announced the official joining with Red Hat while staying independent from RHEL, under a new CentOS governing board.

Source: https://en.wikipedia.org/wiki/CentOS

Datasheet	
Developer	The CentOS Project (affiliated with Red Hat)
Written in	
OS family	Linux Family
Working state	Current
Source model	Open Source
Initial release	14 May 2004
Latest release	8.2.2004[2] (15 June 2020; 5 months ago) [±], 7.9-2009[3] (12 November 2020; 11 days ago) [±], 6.10[4] (3 July 2018; 2 years ago)
Available in	
Platforms	x86-64[a], and since CentOS 8, POWER8 and 64-bit ARM
Kernel type	Linux kernel
Default user interface	Bash, GNOME Shell
License	GNU GPL and other licenses
Official Website	https://centos.org/

Chromium OS

Return to Operating Systems

Chromium OS is a free and open-source operating system designed for running web applications and browsing the World Wide Web. It is the development version of Chrome OS, a Linux distribution made by Google.

Like Chrome OS, Chromium OS is based on the Linux kernel, but its principal user interface is the Chromium web browser rather than the Google Chrome browser. Chromium OS also includes the Portage package manager, which was originally developed for Gentoo Linux. Because Chromium OS and Chrome OS use a web browser engine for the user interface, they are oriented toward web applications rather than desktop applications or mobile apps.

Source: https://en.wikipedia.org/wiki/Chromium_OS

Datasheet	
Developer	Google
Written in	C, C++, Rust, HTML, JavaScript
OS family	Linux
Working state	Current
Source model	Open Source
Initial release	
Latest release	
Available in	
Platforms	x86, x64, ARM, ARM64
Kernel type	Monolithic (Linux kernel)
Default user interface	Chromium (web browser), Aura Shell (Ash)
License	Various open source licenses (mainly BSD-style licenses and GPL)
Official Website	https://www.chromium.org/chromium-os

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:chromium_os

Last update: **2020/12/02 16:27**

Cisco Digital Network Architecture (Cisco DNA)

Return to Operating Systems

Cisco® Digital Network Architecture (Cisco DNA) is your team’s bridge to an intent-based network. It is an open, extensible, software-driven architecture that accelerates and simplifies your enterprise network operations, while lowering costs and reducing your risk. Only Cisco provides a single network fabric that is powered by deep intelligence and integrated security to deliver automation and assurance across your entire organization at scale. Cisco DNA gives IT time back from time-consuming, repetitive network configuration tasks so you can focus on the innovation your business needs.

Cisco DNA Automation and Assurance are built on a Software-Defined Networking (SDN) controller, rich contextual analytics, network virtualization and the limitless scalability of the cloud.

This is an entirely new era of networking.

Most Cisco routers, switches and wireless systems shipping today support Cisco DNA now or with a software update. And with new software subscription offers to choose from Cisco DNA Premier, Cisco DNA Advantage, and Cisco DNA Essentials—you can benefit from new innovations activated through software.

Source:
https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/cisco-digital-network-architecture/nb-06-cisco-dna-soln-ovw-cte-en.html

Datasheet	
Developer	Cisco
Written in	
OS family	
Working state	
Source model	
Initial release	
Latest release	
Available in	
Platforms	
Kernel type	
Default user interface	
License	

Cisco Internetwork Operating System (IOS)

Return to Operating Systems

Cisco Internetwork Operating System (IOS) is a family of network operating systems used on many Cisco Systems routers and current Cisco network switches. Earlier, Cisco switches ran CatOS. IOS is a package of routing, switching, internetworking and telecommunications functions integrated into a multitasking operating system. Although the IOS code base includes a cooperative multitasking kernel, most IOS features have been ported to other kernels such as QNX and Linux for use in Cisco products.

Not all Cisco products run IOS. Notable exceptions include ASA security products, which run a Linux-derived operating system, carrier routers which run IOS-XR and Cisco's Nexus switch and FC switch products which run Cisco NX-OS.

Source: https://en.wikipedia.org/wiki/Cisco_IOS

Datasheet	
Developer	Cisco Systems
Written in	
OS family	
Working state	Current
Source model	Closed Source
Initial release	
Latest release	15.8(3)M[1] / January 22, 2019
Available in	
Platforms	The majority of Cisco routers and current Cisco switches
Kernel type	
Default user interface	Command-line interface (CLI)
License	
Official Website	http://www.cisco.com/en/US/products/ps6537/products_ios_sub_category_home.html

Cisco IOS XR

Return to Operating Systems

Cisco IOS XR is a train of Cisco Systems' widely deployed Internetworking Operating System (IOS), used on their high-end Network Converging System (NCS), carrier-grade routers such as the CRS series, 12000 series, and ASR9000 series.

According to Cisco's product literature, IOS XR shares very little infrastructure with the other IOS trains, and is instead built upon a “preemptive, memory protected, multitasking, microkernel-based operating system”. The microkernel was formerly provided by QNX; versions 6.0 onwards use the Wind River Linux distribution.

Source: https://en.wikipedia.org/wiki/Cisco_IOS_XR

Datasheet	
Developer	Cisco Systems
Written in	
OS family	
Working state	
Source model	
Initial release	
Latest release	
Available in	
Platforms	
Kernel type	
Default user interface	
License	

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:cisco_ios_xr

Last update: **2020/12/02 16:29**

Cisco NX-OS

Return to Operating Systems

Cisco NX-OS is a network operating system for the Nexus-series Ethernet switches and MDS-series Fibre Channel storage area network switches made by Cisco Systems. It evolved from the Cisco operating system SAN-OS, originally developed for its MDS switches.

It is based on Wind River Linux and is inter-operable with other Cisco operating systems.[citation needed] The command-line interface of NX-OS is similar to that of Cisco IOS.

Recent NX-OS has both Cisco-style CLI and Bash shell available.

Source: https://en.wikipedia.org/wiki/Cisco_NX-OS

Datasheet	
Developer	Cisco Systems
Written in	
OS family	Wind River Linux
Working state	
Source model	
Initial release	
Latest release	
Available in	
Platforms	
Kernel type	
Default user interface	
License	

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:cisco_nx-os

Last update: **2020/12/02 16:29**

ClearOS

Return to Operating Systems

ClearOS (also known as the **ClearOS System**, formerly **ClarkConnect**) is an operating system marketed by the software company ClearCenter. It is based on CentOS and Red Hat Enterprise Linux, designed for use in small and medium enterprises as a network gateway and network server with a web-based administration interface. It is positioned as an alternative to Windows Small Business Server. ClearOS succeeds ClarkConnect. The software is built by ClearFoundation, and support services can be purchased from ClearCenter. ClearOS 5.1 removes previous limitations to mail, DMZ, and MultiWAN functions.

As of the ClearOS 6.1 release, the distribution is a full-featured operating system for gateway, network and servers built from source packages for Red Hat Enterprise Linux. Source: URI

Datasheet	
Developer	ClearCenter
Written in	
OS family	Unix Like
Working state	Current
Source model	Open Source
Initial release	7.8.0 23 June 2020
Latest release	
Available in	
Platforms	
Kernel type	Monolithic Kernal
Default user interface	Web User Interface
License	GPL and others
Official Site	http://clearos.com/

CloudReady

Return to Operating Systems

CloudReady is a Chromium OS fork called CloudReady aimed at the educational market, with the intention of extending the life of older PCs and laptops. A subsequent version can dual-boot Neverware and the Windows operating system (until v64).

CloudReady is built on Google's open-source operating system Chromium. Neverware customizes CloudReady so it can be installed and used on PC and Macintosh hardware up to 13 years old, making them behave more like a Chromebook.

Although the company began with an exclusive focus on the US K-12 education sector, it announced in October 2017 its intention to use its Series B funding from Google to further expand into the enterprise market.

Source: https://en.wikipedia.org/wiki/Neverware

Datasheet	
Developer	Neverware
Written in	
OS family	
Working state	
Source model	
Initial release	15 Fenruary 2015
Latest release	
Available in	
Platforms	
Kernel type	
Default user interface	
License	
Official Website	https://www.neverware.com/#intro

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:cloudready

Last update: **2020/12/02 16:27**

ExtremeXOS

Return to Operating Systems

ExtremeXOS is the software or the network operating system used in newer Extreme Networks network switches. It is Extreme Networks second generation operating system after the VxWorks based ExtremeWare operating system.

ExtremeXOS is based on the Linux kernel and BusyBox. In July 2008 legal action was taken against Extreme Networks due to alleged violation of the GNU General Public License. Three months later the lawsuit was settled out of court.

Source: https://en.wikipedia.org/wiki/ExtremeXOS

Datasheet	
Developer	Extreme Networks
Written in	
OS family	Unix-like
Working state	Current
Source model	Closed Source and Open Source
Initial release	Version 10.1, February 2004
Latest release	16.1.3.6 Patch4 (21.1.1 Patch2 for Gen2 Devices) / 05.05.2016 (02.05.2016)
Available in	English
Platforms	Extreme Networks, Network switches
Kernel type	Monolithic (Linux)
Default user interface	Command Line Interface (CLI)
License	
Official Website	http://www.extremenetworks.com/products/extreme-xos.aspx

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:extremexos

Last update: **2020/12/02 16:29**

FreeBSD

Return to Operating Systems

FreeBSD is a free and open-source Unix-like operating system descended from the Berkeley Software Distribution (BSD), which was based on Research Unix. The first version of FreeBSD was released in 1993. In 2005, FreeBSD was the most popular open-source BSD operating system, accounting for more than three-quarters of all installed simply, permissively licensed BSD systems.

FreeBSD has similarities with Linux, with two major differences in scope and licensing: FreeBSD maintains a complete system, i.e. the project delivers a kernel, device drivers, userland utilities, and documentation, as opposed to Linux only delivering a kernel and drivers, and relying on third-parties for system software; and FreeBSD source code is generally released under a permissive BSD license, as opposed to the copyleft GPL used by Linux.

The FreeBSD project includes a security team overseeing all software shipped in the base distribution. A wide range of additional third-party applications may be installed using the pkg package management system or FreeBSD Ports, or by compiling source code.

Much of FreeBSD's codebase has become an integral part of other operating systems such as Darwin (the basis for macOS, iOS, iPadOS, watchOS, and tvOS), FreeNAS (an open-source NAS/SAN operating system), and the system software for the PlayStation 3 and PlayStation 4 game consoles.

Source: https://en.wikipedia.org/wiki/FreeBSD

Datasheet	
Developer	The FreeBSD Project
Written in	
OS family	Unix-like
Working state	Curent
Source model	Open Source
Initial release	1 November 1993
Latest release	12.2 (27 October 2020; 30 days ago) [±], 11.4 (16 June 2020; 5 months ago) [±]
Available in	English
Platforms	ARM, IA-32, x86-64, MIPS, PowerPC, 64-bit SPARC, RISC-V
Kernel type	Monolithic Kernel
Userland	BSD
Default user interface	Unix Shell
License	FreeBSD License, FreeBSD Documentation License
Official Website	https://www.freebsd.org/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:freebsd

Last update: **2020/12/02 16:29**

FreeRTOS

Return to Operating Systems

FreeRTOS is AIX (Advanced Interactive eXecutive, pronounced /ˌeɪaɪˈɛks/) is a series of proprietary Unix operating systems developed and sold by IBM for several of its computer platforms. Originally released for the IBM RT PC RISC workstation, AIX now supports or has supported a wide variety of hardware platforms, including the IBM RS/6000 series and later POWER and PowerPC-based systems, IBM System i, System/370 mainframes, PS/2 personal computers, and the Apple Network Server.

AIX is based on UNIX System V with 4.3BSD-compatible extensions. It is one of four commercial operating systems that have versions certified to The Open Group's UNIX 03 standard (the others being macOS, HP-UX and eulerOS).

The AIX family of operating systems debuted in 1986, became the standard operating system for the RS/6000 series on its launch in 1990, and is still actively developed by IBM. It is currently supported on IBM Power Systems alongside IBM i and Linux.

Source: https://en.wikipedia.org/wiki/FreeRTOS

Datasheet	
Developer	Real Time Engineers Ltd.
Written in	
OS family	Real-time operating systems
Working state	Current
Source model	Open Source
Initial release	
Latest release	10.3.1 19 February 2020
Available in	English
Platforms	ARM (ARM7, ARM9, Cortex-M3, Cortex-M4, Cortex-M7, Cortex-A), Atmel AVR, AVR32, HCS12, MicroBlaze, Cortus (APS1, APS3, APS3R, APS5, FPF3, FPS6, FPS8), MSP430, PIC, Renesas H8/S, SuperH, RX, x86, 8052, Coldfire, V850, 78K0R, Fujitsu MB91460 series, Fujitsu MB96340 series, Nios II, Cortex-R4, TMS570, RM4x, Espressif ESP32, RISC-V
Kernel type	Microkernel
Default user interface	
License	MIT
Official Website	http://www.freertos.org/

IBM i

Return to Operating Systems

IBM i is an integrated operating environment developed by IBM, consisting of operating system, database, middleware, and development tools. IBM i runs on IBM Power Systems servers, as do IBM AIX and Enterprise Linux.

It replaced the i5/OS and OS/400 operating systems but maintains application compatibility with both.

IBM designed IBM i as a "turnkey" operating system, requiring little or no on-site attention from IT staff during normal operation. As such, it has been described as "the driverless variant of IT infrastructure." For example, IBM i has a built-in Db2 database which does not require separate installation. Mass storage ("disks") can be RAIDed or mirrored; when either of those options is configured, one or more disks can be replaced without interrupting work. System administration is wizard-driven. Automatic self-care can schedule all common system maintenance, detect many failures and order spare parts and service automatically.

Source: https://en.wikipedia.org/wiki/IBM_i

Datasheet	
Developer	IBM
Written in	
OS family	IBM i
Working state	Current
Source model	Closed source
Initial release	April 2, 2008;
Latest release	7.4 / April 23, 2019;
Available in	English
Platforms	IBM Power Systems
Kernel type	shares many Microkernel (SLIC) and Virtual machine (TIMI) design philosophies
Default user interface	
License	Proprietary
Official Website	http://www.ibm.com/systems/power/software/i/

iOS

Return to Operating Systems

iOS (formerly **iPhone OS**) is a mobile operating system created and developed by Apple Inc. exclusively for its hardware. It is the operating system that powers many of the company's mobile devices, including the iPhone and iPod Touch; it also powered the iPad until the introduction of iPadOS, a derivative of iOS, in 2019. It is the world's second-most widely installed mobile operating system, after Android. It is the basis for three other operating systems made by Apple: iPadOS, tvOS, and watchOS. It is proprietary software, although some parts of it are open source under the Apple Public Source License and other licenses.

Source: https://en.wikipedia.org/wiki/IOS

Datasheet	
Developer	Apple Inc.
Written in	C, C++, Objective-C, Swift, assembly language
OS family	Unix-like, based on Darwin (BSD), iOS
Working state	Current
Source model	Closed, with Open Source Components
Initial release	29 Jun 2007
Latest release	14.2 (18B92) (November 5, 2020) [±] 14.2.1 (18B121) (November 19, 2020) [±] - iPhone 12 and iPhone 12 Pro-only
Available in	40 languages
Platforms	ARMv8-A (iOS 7 and later), ARMv7-A (iPhone OS 3 – iOS 10.3.4), ARMv6 (iPhone OS 1 – iOS 4.2.1)
Kernel type	Hybrid (XNU)
Default user interface	Cocoa Touch (multi-touch, GUI)
License	Proprietary software except for open-source components
Official Website	https://www.apple.com/ios/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:ios

Last update: **2021/06/14 17:34**

Junos operating system (Junos OS)

Return to Operating Systems

Junos operating system (Junos OS), used in Juniper Networks high-performance network devices creates a responsive and trusted environment for accelerating the deployment of services and applications over a single network.

Junos OS is a single, cohesive operating system that is shared across all network devices and product lines. Junos OS allows Juniper Networks engineers to develop software features once and share these features across all product lines simultaneously, thus reducing the training for each product and interoperability in production environments.

Source: https://en.wikipedia.org/wiki/Junos_OS

Datasheet	
Developer	Juniper Networks
Written in	
OS family	FreeBSD and Linux
Working state	Current
Source model	Closed source and open source
Initial release	7 July 1998
Latest release	0.3RI, 29 September 2020
Available in	English
Platforms	
Kernel type	
Default user interface	Command Line Interface (CLI)
License	Proprietary, Free BSD
Official Website	https://www.juniper.net/us/en/products-services/nos/junos/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:junos_os

Last update: **2021/06/14 17:43**

LynxOS RTOS

Return to Operating Systems

LynxOS RTOS is a Unix-like real-time operating system from Lynx Software Technologies (formerly “LynuxWorks”). Sometimes known as the Lynx Operating System, LynxOS features full POSIX conformance and, more recently, Linux compatibility. LynxOS is mostly used in real-time embedded systems, in applications for avionics, aerospace, the military, industrial process control and telecommunications. As such, it is compatible with military-grade security protocol such as wolfSSL, a popular TLS/SSL library.

Source: https://en.wikipedia.org/wiki/LynxOS

Datasheet	
Developer	Lynx Software Technologies, Inc.
Written in	
OS family	Unix-like real-time operating system
Working state	Current
Source model	Closed Source
Initial release	1986
Latest release	20 February 2020
Available in	
Platforms	Motorola 68010, Intel 80386, ARM architecture, PowerPC
Kernel type	Dynamic Extendable
Default user interface	Command Line Interface (CLI)
License	Proprietary
Official Website	http://www.lynx.com/products/real-time-operating-systems/lynxos-rtos/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:lynxos

Last update: **2020/12/02 16:29**

Nokia X Software Platform

Return to Operating Systems

Nokia X Software Platform was based on the Android Open Source Project (AOSP) and the Linux kernel. Nokia combined Android apps with Nokia experiences (such as HERE Maps, Nokia Xpress and MixRadio) and Microsoft services (such as Skype and Outlook). Nokia officially described the software as bringing “the best of all worlds”. It also encompasses features from the Asha platform, such as the Fastlane notification centre. The user interface mimics that of Windows Phone.

The OS has been compared to Amazon.com's Fire OS, which is also based on AOSP.

Source: https://en.wikipedia.org/wiki/Nokia_X_platform

Datasheet	
Developer	Android Open Source Project (AOSP) code: Google Modifications: Microsoft Mobile (formerly Nokia)
Written in	C (core), C++, Java (UI)
OS family	Linux
Working state	
Source model	Proprietary software based on open source Android and in all devices with proprietary components
Initial release	2014
Latest release	Nokia X software platform 2.1
Available in	
Platforms	32-bit ARM
Userland	Bionic libc, mksh shell, native core utilities with a few from NetBSD
Kernel type	Monolithic (modified Linux kernel)
Default user interface	Graphical (Multi-touch)
License	Proprietary EULA; based on Apache License 2.0 Modified Linux kernel under GNU GPL v2
Official Website	https://developer.nokia.com/nokia-x/platform-overview

Open Network Linux

Return to Operating Systems

Open Network Linux is a Linux distribution for “bare metal” switches, that is, network forwarding devices built from commodity components. ONL uses ONIE to install onto on-board flash memory. Open Network Linux is a part of the Open Compute Project and is a component in a growing collection of open source and commercial projects.

Source: https://www.sdxcentral.com/directory/open-compute-project/open-network-linux/

Datasheet	
Developer	Open Compute Project
Written in	
OS family	Linux
Working state	
Source model	
Initial release	
Latest release	
Available in	
Platforms	
Kernel type	
Default user interface	
License	
Official Website	https://www.sdxcentral.com/directory/open-compute-project/open-network-linux/

OpenServer

Return to Operating Systems

Xinuos **OpenServer**, previously SCO UNIX and SCO Open Desktop (SCO ODT), is a closed source computer operating system developed by Santa Cruz Operation (SCO), later acquired by SCO Group, and now owned by Xinuos. Early versions of OpenServer were based on UNIX System V, while the later OpenServer 10 is based on FreeBSD.

Source: https://en.wikipedia.org/wiki/OpenServer

Datasheet	
Developer	SCO, Caldera Systems, Caldera International, The SCO Group, Xinuos
Written in	
OS family	UNIX System V, BSD
Working state	Current
Source model	Closed SOurce
Initial release	1989
Latest release	10.0 / 2015;
Available in	
Platforms	IA-32, x86-64 (OpenServer 10)
Kernel type	Monolithic Kernel
Default user interface	
License	Proprietary
Official Website	http://www.xinuos.com/menu-products/openserver-10

Oracle Linux (OL)

Return to Operating Systems

Oracle Linux (OL), formerly known as Oracle Enterprise Linux or OEL), is a Linux distribution packaged and freely distributed by Oracle, available partially under the GNU General Public License since late 2006. It is compiled from Red Hat Enterprise Linux (RHEL) source code, replacing Red Hat branding with Oracle's. It is also used by Oracle Cloud and Oracle Engineered Systems such as Oracle Exadata and others.

Oracle Corporation distributes Oracle Linux with two alternative Linux kernels:

- Red Hat Compatible Kernel (RHCK) – identical to the kernel shipped in RHEL
- Unbreakable Enterprise Kernel (UEK[7]) – based on newer mainline Linux kernel versions, with Oracle's own enhancements for OLTP, InfiniBand, SSD disk access, NUMA-optimizations, Reliable Datagram Sockets (RDS), async I/O, OCFS2, and networking.

Oracle promotes Unbreakable Enterprise Kernel as having 100% compatibility with RHEL, even though this is essentially impossible to guarantee due to the kernel's ABI changing due to various factors, including the kernel being based on a newer version which has many thousands of differences from Red Hat's kernel. While the Linux kernel developers, upstream, try never to break userspace, it has happened before. Oracle's compatibility claims lead the user to conclude that third-party RHEL-certified applications will behave properly on the Oracle kernel, but it does not provide any reference to third-party documentation.

Source: https://en.wikipedia.org/wiki/Oracle_Linux

Datasheet	
Developer	Oracle Corporation
Written in	
OS family	Linux (Unix-like)
Working state	current
Source model	Open Source
Initial release	4.5 26 October 2006
Latest release	8.3 13 November 2020
Available in	
Platforms	IA-32, x86-64, SPARC, ARM64
Kernel type	Monolithic Kernel
Default user interface	GNOME and KDE
License	GNU GPL & various others.
Official Website	http://www.oracle.com/linux

Oracle Solaris

Return to Operating Systems

Oracle Solaris is a proprietary Unix operating system originally developed by Sun Microsystems. It superseded the company's earlier SunOS in 1993. In 2010, after the Sun acquisition by Oracle, it was renamed Oracle Solaris.

Solaris is known for its scalability, especially on SPARC systems, and for originating many innovative features such as DTrace, ZFS and Time Slider. Solaris supports SPARC and x86-64 workstations and servers from Oracle and other vendors. Solaris was registered as compliant with UNIX 03 until 29 April 2019.

Source: https://en.wikipedia.org/wiki/Solaris_(operating_system)

Datasheet	
Developer	Sun Microsystems (acquired by Oracle Corporation in 2009)
Written in	C, C++
OS family	Unix
Working state	Current
Source model	Mixed
Initial release	June 1992
Latest release	11.4 / August 28, 2018;
Available in	
Platforms	Current: SPARC, x86-64 Former: IA-32, PowerPC
Kernel type	Monolithic with dynamically loadable modules
Default user interface	GNOME
License	Various
Official Website	https://www.oracle.com/solaris

Red Hat Enterprise Linux (RHEL)

Return to Operating Systems

Red Hat Enterprise Linux (RHEL) is a Linux distribution developed by Red Hat for the commercial market. Red Hat Enterprise Linux is released in server versions for x86-64, Power ISA, ARM64, and IBM Z and a desktop version for x86-64. All of Red Hat's official support and training, together with the Red Hat Certification Program, focuses on the Red Hat Enterprise Linux platform.

The first version of Red Hat Enterprise Linux to bear the name originally came onto the market as “Red Hat Linux Advanced Server”. In 2003, Red Hat rebranded Red Hat Linux Advanced Server to “Red Hat Enterprise Linux AS” and added two more variants, Red Hat Enterprise Linux ES and Red Hat Enterprise Linux WS.

Red Hat uses strict trademark rules to restrict free re-distribution of their officially supported versions of Red Hat Enterprise Linux but still freely provides its source code. Third-party derivatives can be built and redistributed by stripping away non-free components like Red Hat's trademarks. Examples include community-supported distributions like CentOS and Scientific Linux and commercial forks like Oracle Linux. Fedora serves as its upstream source.

Source: https://en.wikipedia.org/wiki/Red_Hat_Enterprise_Linux

Datasheet	
Developer	Red Hat, Inc
Written in	
OS family	Unix Like
Working state	Open Source
Source model	22 February 2000
Initial release	
Latest release	8.3 / October 29, 2020; 7.9 / September 29, 2020;
Available in	Multilingual
Platforms	x86-64; ARM64; IBM Z; IBM Power Systems
Kernel type	Linux
Default user interface	GNOME Shell
License	Various free software licenses, plus proprietary binary blobs
Official Website	https://redhat.com/en/technologies/linux-platforms/enterprise-linux

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:rhel

Last update: **2020/12/02 16:30**

SANtricity Software Operating System (OS)

Return to Operating Systems

SANtricity Software Operating System (OS) performs management tasks while the storage remains online, with complete read and write data access. This capability enables storage administrators to make configuration changes, perform maintenance, or expand storage capacity without disrupting I/O to attached hosts.These online capabilities include the following:

- Dynamic Disk Pools (DDP) technology greatly simplifies traditional storage management with no idle spares to manage or reconfigure when drives are added or fail. This capability enables automatic configuration, expansion, and scaling of storage.
- Dynamic capacity expansion and reduction of DDP pools enable you to add or remove up to 12 drives at a time for a pool. The pool dynamically rebalances to adjust for these drive count changes, with no requirement for parity recalculation.• Dynamic RAID-level migration changes the RAID level of a volume group on the existing drives without requiring the relocation of data. The migration operation supports RAID levels 0, 1, 3, 5, 6, and 10.
- Dynamic physical and logical expansion enables administrators to add new drive modules, configure volume groups, and create volumes without disrupting access to existing data

Source: https://www.netapp.com/pdf.html?item=/media/19775-ds-3171-66862.pdf

Datasheet	
Developer	Nettapp
Written in	
OS family	
Working state	
Source model	
Initial release	
Latest release	
Available in	
Platforms	
Kernel type	
Default user interface	
License	
Official Website	https://www.netapp.com/pdf.html?item=/media/19775-ds-3171-66862.pdf

SCO UnixWare

Return to Operating Systems

SCO UnixWare is a Unix operating system. It was originally released by Univel, a jointly owned venture of AT&T's Unix System Laboratories (USL) and Novell. It was then taken over by Novell. Via Santa Cruz Operation (SCO), it went on to Caldera Systems, Caldera International, and The SCO Group before it was sold to UnXis (now Xinuos). UnixWare is typically deployed as a server rather than a desktop. Binary distributions of UnixWare are available for x86 architecture computers. UnixWare is primarily marketed as a server operating system.[

Source: https://en.wikipedia.org/wiki/UnixWare

Datasheet	
Developer	Univel, Novell, SCO, Caldera Systems, Caldera International, The SCO Group, Xinuos
Written in	
OS family	Unix
Working state	
Source model	Closed Source
Initial release	1992
Latest release	7 Definitive 2018 / 2017
Available in	
Platforms	
Kernel type	Monolithic Kernel
Default user interface	
License	Proproetary
Official Website	https://www.xinuos.com/products/unixware-7/

SUSE Linux Enterprise Server (SLES)

Return to Operating Systems

SUSE Linux Enterprise Server (SLES) is a Linux-based operating system developed by SUSE. It is designed for servers, mainframes, and workstations but can be installed on desktop computers for testing as well. Major versions are released at an interval of 3–4 years, while minor versions (called "Service Packs") are released about every 12 months. SUSE Linux Enterprise products, including SUSE Linux Enterprise Server, receive more intense testing than the upstream openSUSE community product, with the intention that only mature, stable versions of the included components will make it through to the released enterprise product.

It is developed from a common code base with SUSE Linux Enterprise Desktop and other SUSE Linux Enterprise products.

IBM's Watson was built on IBM's POWER7 systems using SLES.

In March 2018, SUSE Product Manager Jay Kruemcke wrote in SUSE blog that SLES developers have ported SUSE Linux Enterprise Server to Raspberry Pi. Source: https://en.wikipedia.org/wiki/SUSE_Linux_Enterprise_Server

Datasheet	
Developer	SUSE
Written in	
OS family	Unix-like
Working state	Current
Source model	Open Source
Initial release	31 August 2000
Latest release	15, 21 July 2020
Available in	Multilingual
Platforms	IA-32 (except SLES 12 and 15), x86-64, s390x, PowerPC, aarch32, aarch64
Kernel type	Monolithic (Linux)
Default user interface	GNOME
Userland	GNU
License	GNU General Public License and various
Official Website	https://www.suse.com/products/server/

TrueNAS

Return to Operating Systems

TrueNAS (called FreeNAS prior to version 12.0) is a free and open-source network-attached storage (NAS) operating system based on FreeBSD and the OpenZFS file system. It is licensed under the terms of the BSD License and runs on commodity x86-64 hardware. TrueNAS supports Windows, macOS and Unix clients and various virtualization hosts such as XenServer and VMware using the SMB, AFP, NFS, iSCSI, SSH, rsync and FTP/TFTP protocols. Advanced TrueNAS features include full-disk encryption and a plug-in architecture for third-party software.

Source: https://en.wikipedia.org/wiki/TrueNAS

Datasheet	
Developer	iXsystems
Written in	
OS family	FreeBSD
Working state	
Source model	
Initial release	
Latest release	TrueNAS-12.0-RELEASE, 20 October 2020
Available in	
Platforms	x86-64, v9.2.1.9 was the last release that supported 32-bit
Kernel type	
Default user interface	
License	BSD License
Official Website	http://truenas.com/

Ubuntu Linux

Return to Operating Systems

Ubuntu Linux is a Linux distribution based on Debian and mostly composed of free and open-source software. Ubuntu is officially released in three editions: Desktop, Server, and Core for Internet of things devices and robots. All the editions can run on the computer alone, or in a virtual machine. Ubuntu is a popular operating system for cloud computing, with support for OpenStack. Ubuntu's default desktop has been GNOME, since version 17.10.

Ubuntu is released every six months, with long-term support (LTS) releases every two years. As of 22 October 2020, the most recent long-term support release is 20.04 (“Focal Fossa”), which is supported until 2025 under public support and until 2030 as a paid option. The latest standard release is 20.10 (“Groovy Gorilla”), which is supported for nine months.

Ubuntu is developed by Canonical, and a community of other developers, under a meritocratic governance model. Canonical provides security updates and support for each Ubuntu release, starting from the release date and until the release reaches its designated end-of-life (EOL) date. Canonical generates revenue through the sale of premium services related to Ubuntu.

Ubuntu is named after the Nguni philosophy of ubuntu, which Canonical indicates means “humanity to others” with a connotation of “I am what I am because of who we all are”.

Source: https://en.wikipedia.org/wiki/Ubuntu

Datasheet	
Developer	Canonical Ltd.
Written in	
OS family	Linux
Working state	Current
Source model	Open Source
Initial release	20 October 2004
Latest release	Ubuntu 20.10 / 22 October 2020
Available in	More than 55 languages
Platforms	IA-32, x86-64, ARM64, ARMhf (ARMv7 + VFPv3-D16), ppc64le (POWER8 and later), s390x
Kernel type	Linux Kernel
Userland	GNU
Default user interface	GNOME
License	Free software + some proprietary device drivers
Official Website	https://ubuntu.com/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:ubuntu_linux

Last update: **2020/12/02 16:30**

Windows

Return to Operating Systems

Microsoft Windows, commonly referred to as **Windows**, is a group of several proprietary graphical operating system families, all of which are developed and marketed by Microsoft. Each family caters to a certain sector of the computing industry. Active Microsoft Windows families include Windows NT and Windows IoT; these may encompass subfamilies, (e.g. Windows Server or Windows Embedded Compact) (Windows CE). Defunct Microsoft Windows families include Windows 9x, Windows Mobile and Windows Phone.

As of October 2020, the most recent version of Windows for PCs, tablets and embedded devices is Windows 10, version 20H2. The most recent version for server computers is Windows Server, version 20H2.[7] A specialized version of Windows also runs on the Xbox One video game console

Source: https://en.wikipedia.org/wiki/Microsoft_Windows

Datasheet	
Developer	Microsoft
Written in	
OS family	Microsoft Windows
Working state	
Source model	Closed-source, Source-available (through Shared Source Initiative)
Initial release	20 November 1985
Latest release	Windows 10, version 20H2
Available in	
Platforms	IA-32, x86-64, ARM, ARM64, Previously: 16-bit x86, DEC Alpha, MIPS, PowerPC, Itanium
Kernel type	
Default user interface	Windows Shell
License	Proprietary commercial software
Official Website	http://microsoft.com/windows

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:windows

Last update: **2020/12/02 16:29**

Windows IoT

Return to Operating Systems

Windows IoT, formerly **Windows Embedded**, is a family of operating systems (OSs) from Microsoft designed for use in embedded systems. Microsoft currently has three different subfamilies of operating systems for embedded devices targeting a wide market, ranging from small-footprint, real-time devices to point of sale (POS) devices like kiosks. Windows Embedded operating systems are available to original equipment manufacturers (OEMs), who make it available to end users preloaded with their hardware, in addition to volume license customers in some cases.

In April 2018, Microsoft released Azure Sphere, another operating system designed for IoT applications running on the Linux kernel.

Source: https://en.wikipedia.org/wiki/Windows_IoT

Datasheet	
Developer	Microsoft
Written in	
OS family	Microsoft Windows
Working state	
Source model	Closed-source, Source-available (through Shared Source Initiative)
Initial release	
Latest release	
Available in	
Platforms	
Kernel type	Hybrid Kernel
Default user interface	
License	Commercial proprietary software
Official Website	https://developer.microsoft.com/en-us/windows/iot

Windows NT

Return to Operating Systems

Windows NT is a family of operating systems produced by Microsoft, the first version of which was released on July 27, 1993. It is a processor-independent, multiprocessing and multi-user operating system.

The first version of Windows NT was Windows NT 3.1 and was produced for workstations and server computers. It was intended to complement consumer versions of Windows that were based on MS-DOS (including Windows 1.0 through Windows 3.1x). Gradually, the Windows NT family was expanded into Microsoft's general-purpose operating system product line for all personal computers, deprecating the Windows 9x family.

“NT” formerly expanded to “New Technology” but no longer carries any specific meaning. Starting with Windows 2000,[4] “NT” was removed from the product name and is only included in the product version string.[5]

NT was the first purely 32-bit version of Windows, whereas its consumer-oriented counterparts, Windows 3.1x and Windows 9x, were 16-bit/32-bit hybrids. It is a multi-architecture operating system. Initially, it supported several instruction set architectures, including IA-32, MIPS, and DEC Alpha; support for PowerPC, Itanium, x64, and ARM were added later. The latest versions support x86 (including IA-32 and x64) and ARM. Major features of the Windows NT family include Windows Shell, Windows API, Native API, Active Directory, Group Policy, Hardware Abstraction Layer, NTFS, BitLocker, Windows Store, Windows Update, and Hyper-V.

Source: https://en.wikipedia.org/wiki/Windows_NT

Datasheet	
Developer	Microsoft
Written in	C, C++, and Assembly language
OS family	
Working state	Deprecated by Windows 2000 30 June 2002
Source model	Closed-source, Source-available (through Shared Source Initiative)
Initial release	Windows NT 3.1, July 27, 1993
Latest release	20H2 10.0.19042.662 (30 November2020; Windows NT Service Pack 4, 17 April 2018
Available in	
Platforms	IA-32, x86-64 and ARM (and historically DEC Alpha, Itanium, MIPS, and PowerPC)
Kernel type	
Default user interface	GUI and Windows Shell
License	Depending on version, edition or customer choice: Trialware, commercial software, volume licensing, OEM-only, SaaS, S+S[a]
Official Website	http://www.microsoft.com/windows/

Windows Server

Return to Operating Systems

Windows Server is a brand name for a group of server operating systems released by Microsoft. It includes all Windows operating systems that are branded “Windows Server”, but not any other Microsoft product. The first Windows server edition to be released under that brand was Windows Server 2003. However, the first server edition of Windows was Windows NT 3.1 Advanced Server, followed by Windows NT 3.5 Server, Windows NT 3.51 Server, Windows NT 4.0 Server, and Windows 2000 Server. Windows 2000 Server was the first server edition to include Active Directory, DNS Server, DHCP Server, Group Policy, as well as many other popular features used today.

Users of Windows Server may choose to deploy either on-site or using a cloud computing service. Each provides different advantages.

By delegating the managing and upkeep of the server to a cloud computing service such as Microsoft Azure or Amazon Web Services, users get the benefit of paying monthly based on usage rather than a large fixed cost. Furthermore, infrastructure tends to be more reliable and it is easier to scale up as necessary. However, buying and running a server in-house may be a better choice in certain cases when it is more cost effective. Other use cases such as using a Windows server to manage client computers in a facility are also appropriate for running a physical server.

Source: https://en.wikipedia.org/wiki/Windows_Server

Datasheet	
Developer	Microsoft
Written in	
OS family	
Working state	Current
Source model	Closed-source, Source available through Shared Source initiative
Initial release	April 24, 2003;
Latest release	version 20H2, Windows Server, version 2004 (10.0.19041) / 27 May 2020;
Available in	
Platforms	
Kernel type	
Default user interface	PowerShell (Command line), Windows shell (Graphical, optional)
License	Trialware, SaaS or volume licensing
Official Website	http://windowsserver.com/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.d_opsys:windows_server

Last update: **2020/12/02 16:30**

Windows XP

Return to Operating Systems

Windows XP is an operating system produced by Microsoft as part of the Windows NT family of operating systems. It was the successor to both Windows 2000 for professional users and Windows Me for home users. It was released to manufacturing on August 24, 2001, and broadly released for retail sale on October 25, 2001.

Development of Windows XP began in the late 1990s as “Neptune”, an operating system (OS) built on the Windows NT kernel which was intended specifically for mainstream consumer use. An updated version of Windows 2000 was also originally planned for the business market; however, in January 2000, both projects were scrapped in favor of a single OS codenamed “Whistler”, which would serve as a single OS platform for both consumer and business markets. As such, Windows XP was the first consumer edition of Windows not to be based on MS-DOS.

Upon its release, Windows XP received critical acclaim, with critics noting increased performance and stability (especially in comparison to Windows Me), a more intuitive user interface, improved hardware support, and expanded multimedia capabilities. However, some industry reviewers were concerned by the new licensing model and product activation system.

Extended support for Windows XP ended on April 8, 2014, after which the operating system ceased receiving further support or security updates (with exceptional security updates being made e.g. in 2019, to address potential ransomware threats, like BlueKeep) to most users. By August 2019, Microsoft (and others) had ended support for games on Windows XP. As of October 2020, 0.8% of Windows PCs run Windows XP, and 0.29% of all devices across all platforms run Windows XP. A few countries still have double-digit use, e.g. Armenia, where it's being replaced by Windows 10, with both operating systems having over 40% use

Source: https://docs.microsoft.com/en-us/windows/win32/srvnodes/windows-server

Datasheet	
Developer	Microsoft
Written in	
OS family	Microsoft Windows
Working state	Deprecated
Source model	Closed-source, Source-available (through Shared Source Initiative)
Initial release	24 August 2001
Latest release	Service Pack 3 (5.1.2600), 21 April 2008; Some securoty updates available up to 2019, especially for embedded XP
Available in	
Platforms	IA-32, x86-64, and Itanium
Kernel type	
Default user interface	
License	

Official Website	

Appendix E: Tools

Return to Reference Architecture (RA) or Return to Appendices

As with all projects, DIDOs require tools to support their development, management, and operation. The following lists of tools are meant to be informative and by no means exhaustive.

Community Tools

The following tools are used to aid in the development of software, particularly the development of Open Source Software (OSS).

- Tools: Archiving and Release Management
- Tools: Bug and Issue Tracking
- Tools: Code Reviews
- Tools: Contributor License Agreements (CLA)
- Tools: GitHub Management at Corporate Scale
- Tools: Logging Tools
- Tools: Open Source Paradigm
- Tools: Project Quality
- Tools: Source Code Scanning and License Compliance
- Tools: Tracking Project Health

Network Traffic Analysis Tools

- Tools: Network Traffic Analysis

Tools for communications and collaboration

- Tools TBD

Tools for corporate-scale GitHub management

- Tools TBD

Tools: Archiving and Release Management

Return to Tools Area

Source: Tools for managing open source programs

- ***Artifactory*** *– Artifactory is a repository manager from JFrog which supports software packages created in any code language. It integrates with all major DevOps and continuous integration and continuous deployment tools. Artifactory is available as a commercial or as an open source tool. https://www.jfrog.com/artifactory/*
- ***Bintray*** *– A commercial archiving tool from JFrog that allows companies to publish their code release archives to maintain storage for older and larger files. https://bintray.com/*
- ***Docker Hub*** *– A cloud-based registry service which allows users to link to code repositories and build and test their images. It also stores manually-pushed images and links to Docker Cloud so users can deploy images to project hosts. Docker Hub is a centralized resource for container image discovery, distribution and change management, collaboration and workflow automation throughout the development pipeline. https://hub.docker.com/*
- ***github-release*** *– The open source, built in functionality part of GitHub which lets users package and edit releases of projects on GitHub so they are available for use by other community members. https://github.com/aktau/github-release*

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.e_tools:release-mgmt

Last update: **2021/06/09 22:06**

Tools: Bug and Issue Tracking

Return to Tools Area

Source: Tools for managing open source programs

- ***Bugzilla*** *– Open source, server-based software featuring an advanced query tool that can remember searches, integrated email capabilities and a comprehensive permissions system. Bugzilla is used by Mozilla as its bug tracking system.https://www.bugzilla.org/*
- ***GitHub Issues*** *– GitHub's own integrated feedback and bug tracker, GitHub Issues is available as part of GitHub's project hosting.https://help.github.com/articles/about-issues/*
- ***GitLab*** *– This bug tracking tool unifies issue tracking, code review, Git repository management, activity streams, wikis and more in a single UI to assist your open source projects. GitLab is available as a service or as a commercial software.https://about.gitlab.com/*
- ***JIRA*** *– From Atlassian, JIRA contains custom filters, developer tool integrations, customizable workflows and rich APIs to integrate JIRA with other applications. JIRA is available as a commercial software. https://www.atlassian.com/software/jira*

Tools: Code Reviews

Return to Tools Area

Source: Tools for managing open source programs

- ***mention-bot*** *– Developed by Facebook, this tool automatically mentions potential reviewers for code contributions by community members to speed up the review process.https://github.com/facebook/mention-bot*
- ***PullApprove*** *– Brings more formalization to code contributions – or pull requests – by improving code quality through peer-review, enforcing style guidelines, catching errors and providing security checks on code.https://about.pullapprove.com/*
- ***sentinel*** *– A repository management bot which reviews and tests code contributions, builds a list of maintainers for the repository and communicates the status of a pull request with users.https://github.com/habitat-sh/sentinel*

Tools: Contributor License Agreements (CLA)

Return to Tools Area

***Contributor License Agreements (CLA)** is legal documentation that serves to protect any contributors to a project in regard to copyright issues and distribution. Essentially the CLA states that the contributor writer/computer programmer/etc. has agreed to allow their input to be used in the ongoing development of the project, while protecting them against future copyright infringement, and permitting them to take legal action should there be a case of infringement or should they not be receiving credit for their contributions. Additionally, it protects the other parties, as well, as the CLA does not allow for the contributor to later refuse use of their material.* *https://www.upcounsel.com/contributor-license-agreement*

There are different tools to help with CLA[53]

- ***CLA Assistant** – Contributed by SAP, the CLA Assistant streamlines workflows by handling the legal side of contributions for users. The Assistant asks code contributors to sign CLAs as they make their code contributions and authenticates each contributor with his or her GitHub account. It also updates the status of a pull request when the contributor agrees to the CLA and automatically asks users to re-sign the CLA for each new pull request if changes are made to the CLA.https://github.com/cla-assistant/cla-assistant*
- ***CLA Portal** – From VMware, CLA Portal adds a workflow to enable contributors to digitally sign a CLA for pull requests to your GitHub repositories. When a developer opens a pull request, they are prompted to sign the agreement if needed. Also included is an administrator interface for CLA authoring, CLA-to-project mapping, and agreement reviews.https://github.com/vmware/claportal*
- ***DCOB** – A Developer Certificate of Origin Bot which helps to enforce developer certificate of origin sign-offs for each code change in a pull request. The DCOB sets the status for each accepted code change, as required by the Developer Certificate of Origin.https://github.com/chef/dcob*

[53] Tools for managing open source programs

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.e_tools:code-signing

Last update: **2021/06/09 22:06**

Tools: GitHub Management at Corporate Scale

Return to Tools Area

Source: Tools for managing open source programs

- ***hubcommander*** *- A Slack bot for GitHub organization management, HubCommander uses chat-ops – or conversation-driven development – to help manage GitHub projects. It creates a simple way to perform privileged GitHub organization management tasks without granting administrative or owner privileges to your GitHub organization members.https://github.com/Netflix/hubcommander*
- ***opensource-portal*** *– From Microsoft, this tool is designed to help large organizations with their large-scale GitHub management operations, onboarding and more. This is one of a suite of tools provided by the Open Source Programs Office at Microsoft.https://github.com/Microsoft/opensource-portal*
- ***settings*** *– This app syncs repository settings defined in .github/settings.yml to GitHub, enabling pull requests for repositories.https://github.com/bkeepers/github-configurer*
- ***zappr*** *- Zappr is a GitHub integration built to enhance project workflows. From Zalando, zappr helps developers to increase productivity and improve open-source project quality by removing bottlenecks around pull request approval and helping project owners halt “rogue” pull requests before they’re merged into the project master branches.https://github.com/zalando/zappr*

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.e_tools:project-oversigt

Last update: **2021/06/09 22:07**

Tools: Logging Tools

Return to Tools Area

Source: Apache Logging Services

- ***Apache Chainsaw*** *is GUI based log viewer and is a companion application to log4j written by members of the log4j development community. Chainsaw can read local and ssh-reachable regular text log files, as well as log files formatted in Log4j's XMLLayout. Chainsaw can also receive events over TCP and UDP, read events from a DB, and can process events generated by java.util.logging. https://logging.apache.org/chainsaw.*
- ***Apache Log4j Audit*** *provides the information to determine what actions have been performed, or attempted to be performed, by whom, when they did it and what the data associated with the action was. Audit log data is most typically used by security teams to monitor for fraud or illegal or otherwise unauthorized activities and to be able to correct changes that were incorrectly made. Audit logs are often used to demonstrate compliance with legal obligations such as Sarbanes-Oxley Act or HIPPA.https://logging.apache.org/log4j-audit/latest/*

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.e_tools:logging

Last update: **2021/06/09 22:07**

Tools: Network Traffic Analysis

Return to Tools Area

Network Traffic Analyzers are generally deployed within Enterprises. Although having many of the characteristics of a single, united Enterprise, DIDOs are naturally more of a federation or coalition of the willing spread across the internet. This is why virtualized testing is so important.

Source: The 14 Best Network Traffic Analysis Solutions for 2019 and Beyond, September 17, 2019, https://solutionsreview.com/network-monitoring/the-14-best-network-traffic-analysis-solutions-for-2019-and-beyond/

- ***Awake Security Platform*** *is a network traffic analysis solution that focuses on discovering, assessing, and processing security threats. The tool is broken down into three parts: Awake Sensors, which continuously monitor and collect data from devices, apps, and users; Awake Nucleus, which analyzes that data to understand behaviors and attributes of entities and applying deep forensics; and Ava, a privacy-aware security expert system that applies machine learning to collected data.*
- ***Corelight*** *is a security-focused network traffic analysis provider that uses the open source network security monitor Zeek as its basis. Corelight Sensors convert network traffic data into logs and extracted files which can all be managed through the Corelight Fleet Manager. Through the Fleet Manager, admins can define custom groups, assign individual roles, and set access levels. Corelight Sensors come either as hardware for networks, as a virtual sensor, or as a cloud traffic monitor for AWS.*
- ***Flowmon*** *is a network performance and security solution provider that offers network traffic monitoring and analysis capabilities. The solution offers real-time NetFlow and IPFIX monitoring and analyzes network traffic data from a physical, virtual, or cloud infrastructure. It also gathers flow data statistics generated by routers, switches, or standalone hardware probes. Users can add self-defined filters that set parameters for data collection based on what data the user wants to look at.*
- ***Kentik Platform*** *is an AIOps platform that applies artificial intelligence and machine learning capabilities to network traffic analysis. The solution analyzes downstream and transit traffic flows and helps enterprises identify peering opportunities, optimize their network routing, and gain more control over their service performance. They also offer network traffic engineering capabilities to maximize resource utilization and traffic delivery, and insights into network capacity to help drive cost-efficient traffic flow.*
- ***LogRhythm NetworkXDR*** *is a security-focused network traffic analysis solution that focuses on threat detection and analytics. It offers real-time network traffic analysis via network sensors that allow for distributed traffic data collection and reporting. The solution is designed to increase network traffic visibility with application identification, app-aware metadata, and full packet capture. NetworkXDR also integrates with LogRhythm's NextGen SIEM Platform to help identify security threats.*
- ***ManageEngine Netflow Analyzer*** *is a bandwidth monitoring tool that is built on network traffic monitoring and analysis functions. The program implements network flow analysis to examine*

bandwidth usage, network data, and traffic patterns. It condenses information about which users and devices are using available bandwidth on your network – as well as what they're using it for. The solution also feature network forensics and security features, application monitoring, and data capacity planning and billing capabilities.

- ***Netfort LANGuardian** is a network traffic analysis and packet inspection software that monitors network and user activity. LANGuardian uses packet inspection tools to troubleshooting bandwidth problems, create audit trails of file and folder activity, and examine Internet gateways. The solution uses wire data analytics to capture metadata from network packets, provides continuous health checks on network and user activity, and alerts admins to any suspicious data.*

- ***NETSCOUT** is a service assurance and network monitoring vendor that provides network traffic data inspection and analysis. The solution continuously inspects traffic data and analyzes large volumes of data through Layer 7/8 deep packet inspection, load balancing and acceleration, aggregation and desegregation, and packet decoding. NETSCOUT also utilizes their Adaptive Service Intelligence (ASI) technology that uses traffic data to gain visibility into user communities, services, and IT assets.*

- ***ntopng** is an open source network traffic probe and analysis tool. The traffic probe sorts network traffic into different criteria, including IP addresses and throughput. By characterizing network traffic, your enterprise can easily determine different network statistics that are affecting your network; the solution can reference real-time and historical traffic data in this analysis. While ntopng's Community version is open source, Professional and Enterprise versions are also available.*

- ***Paessler PRTG** is an IT monitoring tool that includes network traffic analysis functionality. PRTG's network traffic analysis system helps administrators track network capacity and seeing how much of their data analysis is actually being used. The solution combines SNMP monitoring, packet sniffing, and data flow technologies like NetFlow, IPFIX, jFlow, and sFlow for their traffic analysis capabilities; it displays traffic data alongside the other performance and security insights it uncovers.*

- ***Plixer Scrutinizer** is a network traffic analysis system that gathers network traffic flow and metadata across an entire network infrastructure. The solution collects data from SD-WAN, cloud, firewalls, routers, data centers, probes, data collectors, and wired/wireless edges. Scrutinizer then takes this data and provides valuable security and performance insights. This tool can help IT teams optimize network and application performance by providing end-to-end network visibility.*

- ***SolarWinds NetFlow Traffic Analyzer** is a NetFlow traffic analysis and bandwidth monitoring solution. The tool is designed specifically to analyze NetFlow traffic data as well as IPv4 and IPv6 flow records and application traffic. Users can also visually correlate performance and traffic data discrepancies by displaying metrics right next to each other. It also can integrate with SolarWinds' other Orion Platform products, such as their Network Performance Monitor and Network Configuration Manager.*

- ***Ipswitch WhatsUp Gold** is an all-in-one infrastructure monitoring tool that features network traffic analysis capabilities. WhatsUp Gold provides insight into application bandwidth usage and helps administrators to manage the performance of your infrastructure, applications, and services. It also leverages real-time and historical bandwidth usage data to help enterprises keep track of capacity, as well as determine what traffic was consuming bandwidth during a period of slow* *network performance.*

Tools: Open Source Paradigm

Return to Tools Area

Using a DIDO is not just a simple shift in policies, procedures and practices. It is a change in the architectural paradigm – from centralized control to distributed – requiring a fundamental change in how a system is normalized into systems, subsystems, components, etc. It also requires a shift in the basic underlying principles of a system. DIDOs are generally:

- Comprised of thousands, if not millions, of independent nodes
- Outside the control of any one individual or corporation
- Lacking centralized authority, with decisions made by consensus

The DIDO architecture does not represent a single unified enterprise; rather it represents a loosely defined confederation of domains bound together by systems integration (SI)[34]. Although SI is not a new concept to enterprises, the granularity and kinds components associated with a DIDO architecture requires a rethink. Within the DIDO environment, the definition of a platform shifts from hardware, operating system, software languages, and services (i.e., web, app, database, etc.) components to the DIDO Platform components. It is the responsibility of the DIDO Platform to isolate the enterprise from the traditional platform concerns.

The granularity of the data elements within an enterprise can also shift to smaller more isolated objects, which represent only a portion of the traditional Data Model (DM). In other words, the enterprise's data model is not going to be deployed into a single DIDO, nor should it. Enterprise data stores will continue and will be augmented by the DIDO. Some data will reside completely within the Enterprise data stores, some data will reside completely within the DIDO, and some data will straddle both. Data that straddles both will need more procedures and policies to ensure data integrity.

Relevant Open Source Standards

The cultural shift from a stove-piped corporate or enterprise culture with almost complete control to one with a systems integrator participating in numerous distributed communities covering a wide range of domains requires committed leadership and a concerted effort by all the players involved.

Technical Standards

- None at this time.

de facto Standards

- There are none at this time, but there are guides on participating for Open Source initiatives. There are many written on this subject. **Talk Openly Develop Openly** (TODO) provides an extensive reading list.[35]. TODO also provides the following excellent guide as a place to start.

- TODO: Participating in open source communities

[34] **System Integrator** - An individual or organization that builds systems from a variety of diverse components. With increasing complexity of technology, more customers want complete solutions to information problems, requiring hardware, software and networking expertise in a multivendor environment. https://www.pcmag.com/encyclopedia/term/52450/systems-integrator

[35] TODO Open Source Reading List, https://todogroup.org/guides/open-source-reading-list/

Tools: Project Quality

Return to Tools Area

Source: Tools for managing open source programs

- ***CII Best Practices Badging*** *– From The Linux Foundation, the Core Infrastructure Initiative (CII) Best Practices badge is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify for free by using this web application to explain how they follow each best practice.https://bestpractices.coreinfrastructure.org/*
- ***CodeClimate*** *– Code Climate empowers organizations to take control of their code quality by incorporating fully configurable test coverage and maintainability data throughout the development workflow. It's free for open source projects!https://codeclimate.com/*

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.e_tools:project-quality

Last update: **2021/06/09 22:08**

Tools: Source Code Scanning and License Compliance

Return to Tools Area

Source: Tools for managing open source programs

- ***Antepedia Reporter*** *– A commercial, fee-based application from Antepedia, Reporter is a report-generation product which lets developers, project managers, legal advisors and others create license compliance audits and Intellectual Property (IP) rights management reports about the open source, public and private components in your code base. http://www.antepedia.com/pages/tools.html*
- ***Black Duck Hub*** *– The commercial Hub service scans code to identify all embedded open source components, and then automatically searches for known vulnerabilities for remediation. It can send alerts when new vulnerabilities are found in your code. https://www.blackducksoftware.com/products/hub*
- ***Black Duck Protex*** *– Protex is a commercial, fee-based license compliance management tool from Black Duck which integrates with existing tools to automatically scan, identify and inventory open source software, while also enforcing license compliance and corporate policy requirements.https://www.blackducksoftware.com/products/protex*
- ***Copyright review tools*** *– This collection of open-source command-line tools help make initial copyright file construction and subsequent review and update easier. https://wiki.debian.org/CopyrightReviewTools*
- ***dep-checker*** *– A free dependency checker tool from The Linux Foundation, dep-checker performs a complete analysis of linkages between code packages. http://git.linuxfoundation.org/dep-checker.git/*
- ***FlexNet Code Insight*** *– Flexera, which acquired licensing compliance vendor Palamida in 2016, commercially offers Flex Code Insight to help automate corporate open source use among developers, legal teams and security staffers. https://www.flexerasoftware.com/enterprise/products/software-vulnerability-management/flexnet-code-insight/*
- ***FOSSA*** *– This is a commercial tool that automatically performs code dependency tracking, license compliance scanning in the background. http://fossa.io/*
- ***FOSSology*** *– A Linux Foundation project, FOSSology is an open-source license compliance software toolkit that can run license, copyright and export control scans from the command line. A database and web UI are also available to create compliance workflows. https://www.fossology.org/*
- ***janitor.git*** *– Code Janitor is an open-source tool that helps evaluate source code for compliance with open source licenses. From The Linux Foundation, Code Janitor can be used with other products to check code. http://git.linuxfoundation.org/janitor.git/*

- ***LicenseFinder*** *– An open-source tool that detects the licenses of the code being used in your projects, compares those licenses against a user-defined whitelist and then provides an actionable report. https://github.com/pivotal/LicenseFinder*

- ***Protecode Enterprise Analyzer*** *– This commercial application is used to analyze and identify all code in any directory to determine code ownership and ensure open source license compliance based on predetermined internal policies. http://www.protecode.com/our-products/system-4/enterprise-analyzer/*

- ***scancode-toolkit*** *– From nexB, the open source ScanCode suite of utilities scans code for licenses, copyright, and dependencies to find, discover and inventory open source and third-party components used in your code. https://github.com/nexB/scancode-toolkit*

- ***SPDX*** *– The Software Package Data Exchange (SPDX) specification is a standard format used to describe the components, licenses, and Copyrights associated with software packages. The SPDX standard aids compliance with free and open-source software licenses by standardizing the way license information is shared between developers and companies. The SPDX specification is developed by the SPDX workgroup, which is hosted by The Linux Foundation. The group offers open-source tools to help users of SPDX documents. https://spdx.org/tools*

- ***WhiteSource*** *– Provides licensing, security, code quality, and reporting analysis for managing open source components in real-time by automatically and continuously scanning dozens of open source repositories on a commercial basis. https://www.whitesourcesoftware.com/*

Tools: Tracking Project Health

Return to Tools Area

Source: Tools for managing open source programs

- ***CatWatch*** – *CatWatch is an open source metrics dashboard from Zalando that fetches GitHub statistics for your GitHub accounts, processes and saves your GitHub data in a database. The data reveals the popularity of your open source projects, your most active contributors and other interesting statistics.https://github.com/zalando-incubator/catwatch*
- ***Gander*** – *Gander is an open source dashboard which generates usable metrics for a range of open source projects in one quick look. Created by PayPal, Gander is designed for individuals who are responsible for running Open Source Program Offices or keeping track of multiple open source projects. https://github.com/paypal/Gander*
- ***GHCrawler*** – *Created by Microsoft, GHCrawler is an open source GitHub API crawler that crawls a GitHub-hosted project and automatically tracks, retrieves, and stores its contents. GHCrawler is primarily intended for people trying to track sets of organizations and data repositories. https://github.com/Microsoft/ghcrawler*
- ***Gittagstats*** – *Gittagstats is an open source tool which generates statistics reports from a set of tags for a Git repository. The tool was created by Qualcomm. https://github.com/mcharleb/gittagstats*
- ***GrimoireLab*** – *GrimoireLab has a variety of open source tools to measure open source project statistics and visualize them, from git repositories, GitHub pull requests or Bugzilla tickets to mailing lists, Meetup groups or Slack channels. GrimoireLab is a project in CHAOSS, a collaborative group on open source development metrics. https://grimoirelab.github.io/*
- ***OSS-dashboard*** – *The Open Source Program Dashboard, which comes from Amazon, is a multi-function dashboard which can be used to view and monitor many GitHub organizations and or users at one time. https://github.com/amzn/oss-dashboard*
- ***OSS Tracker*** – *OSS Tracker, from Netflix, collects data about a GitHub organization and aggregates it across all projects within that organization in a single user interface. All repositories are listed and metrics are combined for an organization, but community managers can also organize projects into functional areas and appoint administrators to assign management and engineering leads. https://github.com/Netflix/osstracker*

Appendix F: DDS Quality Of Service

Return to Reference Architecture (RA) or Return to Appendices

To Be Updated

About

These are the Quality of Service (QoS) parameters defined in the main Data Distribution Service (DDS) specification.

DDS Quality of Service Parameters

The following list of DDS QoS parameters are listed in the order they appear in the DDS Specification rather than in Alphabetical order. (Source: DDS Section 2.2.3 Supported QoS)

1. User Data
2. Topic Data
3. Group Data
4. Durability
5. Durability Service
6. Presentation
7. Deadline
8. Latency Budget
9. Ownership
10. Ownership Strength
11. Liveliness
12. Time Based Filter
13. Partition
14. Reliability
15. Transport Priority
16. Lifespan
17. Destination Order
18. History
19. Resource Limits
20. Entity Factory
21. Writer Data Lifecycle
22. Reader Data Lifecycle

F.1 User Data

Return to DDS Quality of Service

The purpose of USER_DATA QoS is to allow the application to attach additional information to the created Entity objects such that when a remote application discovers their existence it can access that information and use it for its own purposes. One possible use of this QoS is to attach security credentials or some other information that can be used by the remote application to authenticate the source. In combination with operations such as `ignore_participant`, `ignore_publication`, `ignore_subscription`, and `ignore_topic` these QoS can assist an application to define and enforce its own security policies. The use of this QoS is not limited to security, rather it offers a simple, yet flexible extensibility mechanism.

Source: DDS 1.4 Spec

F.2 Topic Data

Return to DDS Quality of Service

The `TOPIC_DATA` QoS allows the application to attach additional information to the created Topic such that when a remote application discovers their existence it can examine the information and use it in an application-defined way. In combination with the listeners on the `DataReader` and `DataWriter` as well as by means of operations such as ignore_topic, these QoS can assist an application to extend the provided QoS.

Source: DDS 1.4 Spec

F.3 Group Data

Return to DDS Quality of Service

The `GROUP_DATA` QoS allows the application to attach additional information to the created `Publisher` or `Subscriber`. The value of the `GROUP_DATA` is available to the application on the `DataReader` and `DataWriter` entities and is propagated by means of the built-in topics.

This QoS can be used by an application combination with the `DataReaderListener` and `DataWriterListener` to implement matching policies similar to those of the `PARTITION QoS` except the decision can be made based on an application-defined policy.

Source: DDS 1.4 Spec

F.4 Durability

Return to DDS Quality of Service

Note

> The decoupling between `DataReader` and `DataWriter` offered by the Publish/Subscribe paradigm allows an application to write data even if there are no current readers on the network. Moreover, a DataReader that joins the network after some data has been written could potentially be interested in accessing the most current values of the data as well as potentially some history.

The DURABILITY QoS policy controls whether the Service will actually make data available to late-joining readers. Note that although related, this does not strictly control what data the Service will maintain internally. That is, the Service may choose to maintain some data for its own purposes (e.g., flow control) and yet not make it available to late-joining readers if the DURABILITY QoS policy is set to VOLATILE.

The value offered is considered compatible with the value requested if and only if the inequality `offered kind >= requested kind` evaluates to TRUE. For the purposes of this inequality, the values of DURABILITY kind are considered ordered such that `VOLATILE < TRANSIENT_LOCAL < TRANSIENT < PERSISTENT`.

For the purpose of implementing the DURABILITY QoS kind TRANSIENT or PERSISTENT, the service behaves “as if” for each `Topic` that has TRANSIENT or PERSISTENT DURABILITY kind there was a corresponding “built-in” `DataReader` and `DataWriter` configured to have the same DURABILITY kind. In other words, it is “as if” somewhere in the system (possibly on a remote node) there was a “built-in durability `DataReader`” that subscribed to that Topic and a “built-in durability `DataWriter`” that published that Topic as needed for the new subscribers that join the system.

For each `Topic`, the built-in fictitious “persistence service” `DataReader` and `DataWriter` has its QoS configured from the `Topic` QoS of the corresponding `Topic`. In other words, it is “as-if” the service first did find_topic to access the `Topic`, and then used the QoS from the Topic to configure the fictitious built-in entities.

A consequence of this model is that the transient or persistence serviced can be configured by means of setting the proper QoS on the Topic.

For a given Topic, the usual request/offered semantics apply to the matching between any DataWriter in the system that writes the Topic and the built-in transient/persistent DataReader for that Topic; similarly for the built-in transient/persistent DataWriter for a Topic and any DataReader for the Topic. As a consequence, a DataWriter that has an incompatible QoS with respect to what the Topic specified will not send its data to the transient/persistent service, and a DataReader that has an incompatible QoS with respect to the specified in the Topic will not get data from it.

Incompatibilities between local `DataReader/DataWriter` entities and the corresponding fictitious “built-in transient/persistent entities” cause the

REQUESTED_INCOMPATIBLE_QOS/OFFERED_INCOMPATIBLE_QOS status to change and the corresponding Listener invocations and/or signaling of `Condition` and `WaitSet` objects as they would with non-fictitious entities.

The setting of the service_cleanup_delay controls when the TRANSIENT or PERSISTENT service is able to remove all information regarding a data-instances. Information on a data-instances is maintained until the following conditions are met:

1. The instance has been explicitly disposed (instance_state = NOT_ALIVE_DISPOSED)
2. While in the NOT_ALIVE_DISPOSED state the system detects that there are no more “live” `DataWrite` entities writing the instance, that is, all existing writers either unregister the instance (call unregister) or lose their liveliness
3. And finally, a time interval longer that service_cleanup_delay has elapsed since the moment the service detected that the previous two conditions were met.

Source: DDS 1.4 Spec

F.5 Durability Service

Return to DDS Quality of Service

The DURABILITY_SERVICE policy configures the HISTORY QoS and the RESOURCE_LIMITS QoS used by the fictitious `DataReader` and `DataWriter` used by the “persistence service.” The “persistence service” is the one responsible for implementing the DURABILITY kinds TRANSIENT and PERSISTENCE.

Source: DDS 1.4 Spec

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:durability_service

Last update: **2021/06/12 16:49**

F.6 Presentation

Return to DDS Quality of Service

The PRESENTATION QoS policy controls the extent to which changes to data-instances can be made dependent on each other and also the kind of dependencies that can be propagated and maintained by the Service.

The setting of coherent_access controls whether the Service will preserve the groupings of changes made by the publishing application by means of the operations begin_coherent_change and end_coherent_change.

The setting of ordered_access controls whether the Service will preserve the order of changes.

The granularity is controlled by the setting of the access_scope.

If coherent_access is set, then the access_scope controls the maximum extent of coherent changes. The behavior is as follows:

- If `access_scope` is set to INSTANCE, the use of `begin_coherent_change` and `end_coherent_change` has no effect on how the subscriber can access the data because with the scope limited to each instance, changes to separate instances are considered independent and thus cannot be grouped by a coherent change.
- If `access_scope` is set to TOPIC, then coherent changes (indicated by their enclosure within calls to `begin_coherent_change` and `end_coherent_change`) will be made available as such to each remote `DataReader` independently. That is, changes made to instances within each individual `DataWriter` will be available as coherent with respect to other changes to instances in that same `DataWriter`, but will not be grouped with changes made to instances belonging to a different `DataWriter`.
- If `access_scope` is set to GROUP, then coherent changes made to instances through a `DataWriter` attached to a common `Publisher` are made available as a unit to remote subscribers.

If `ordered_access` is set, then the access_scope controls the maximum extent for which order will be preserved by the Service.

- If `access_scope` is set to INSTANCE (the lowest level), then changes to each instance are considered unordered relative to changes to any other instance. That means that changes (creations, deletions, modifications) made to two instances are not necessarily seen in the order they occur. This is the case even if it is the same application thread making the changes using the same `DataWriter`.
- If `access_scope` is set to TOPIC, changes (creations, deletions, modifications) made by a single `DataWriter` are made available to subscribers in the same order they occur. Changes made to instances through different `DataWriter` entities are not necessarily seen in the order they occur. This is the case, even if the changes are made by a single application thread using `DataWriter` objects attached to the same `Publisher`.
- Finally, if `access_scope` is set to GROUP, changes made to instances via `DataWriter` entities

attached to the same `Publisher` object are made available to subscribers on the same order they occur.

Note:

> This QoS policy controls the scope at which related changes are made available to the subscriber. This means the subscriber can access the changes in a coherent manner and in the proper order; however, it does not necessarily imply that the Subscriber will indeed access the changes in the correct order. For that to occur, the application at the subscriber end must use the proper logic in reading the `DataReader` objects, as Data Distribution Service (DDS) 1.4 Spec, Access to the data.

The value offered is considered compatible with the value requested if and only if the following conditions are met :

1. The inequality “offered access_scope >= requested access_scope” evaluates to ‘TRUE.’ For the purposes of this inequality, the values of PRESENTATION access_scope are considered ordered such that INSTANCE < TOPIC < GROUP.
2. Requested coherent_access is FALSE, or else both offered and requested coherent_access are TRUE.
3. Requested ordered_access is FALSE, or else both offered and requested ordered _access are TRUE.

Source: DDS 1.4 Spec

F.7 Deadline

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:latency_budget

Last update: **2021/06/12 16:49**

F.8 Latency Budget

Return to DDS Quality of Service

To Be Completed

F.9 Ownership

Return to DDS Quality of Service

To Be Completed

F.10 Ownership Strength

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:ownership_strength

Last update: **2021/06/12 16:50**

F.11 Liveliness

Return to DDS Quality of Service

The LIVELINESS policy controls the mechanism and parameters used by the Service to ensure that particular entities on the network are still “alive.” The liveliness can also affect the ownership of a particular instance, as determined by the OWNERSHIP QoS policy.

This policy has several settings to support both data-objects that are updated periodically as well as those that are changed sporadically. It also allows customizing for different application requirements in terms of the kinds of failures that will be detected by the liveliness mechanism.

The AUTOMATIC liveliness setting is most appropriate for applications that only need to detect failures at the processlevel, but not application-logic failures within a process. The Service takes responsibility for renewing the leases at the required rates and thus, as long as the local process where a `DomainParticipant` is running and the link connecting it to remote participants remains connected, the entities within the `DomainParticipant` will be considered alive. This requires the lowest overhead.

The MANUAL settings (MANUAL_BY_PARTICIPANT, MANUAL_BY_TOPIC), require the application on the publishing side to periodically assert the liveliness before the lease expires to indicate the corresponding Entity is still alive. The action can be explicit by calling the `assert_liveliness` operations, or implicit by writing some data.

The two possible manual settings control the granularity at which the application must assert liveliness.

- The setting MANUAL_BY_PARTICIPANT requires only that one Entity within the publisher is asserted to be alive to deduce all other Entity objects within the same `DomainParticipant` are also alive.
- The setting MANUAL_BY_TOPIC requires that at least one instance within the DataWriter is asserted.

The value offered is considered compatible with the value requested if and only if the following conditions are met:

1. the inequality `offered kind >= requested kind` evaluates to TRUE. For the purposes of this inequality, the values of LIVELINESS kind are considered ordered such that: `AUTOMATIC < MANUAL_BY_PARTICIPANT < MANUAL_BY_TOPIC`.
2. the inequality `offered lease_duration ⇐ requested lease_duration` evaluates to TRUE.

Changes in LIVELINESS must be detected by the Service with a time-granularity greater or equal to the `lease_duration`. This ensures that the value of the LivelinessChangedStatus is updated at least once during each lease_duration and the related Listeners and WaitSets are notified within a `lease_duration` from the time the LIVELINESS changed.

Source: DDS 1.4 Spec

F.12 Time Based Filter

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:time_based_filter

Last update: **2021/06/12 16:50**

F.13 Partition

Return to DDS Quality of Service

To Be Completed

F.14 Reliability

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:reliability

Last update: **2021/06/12 16:51**

F.15 Transport Priority

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:transport_priority

Last update: **2021/06/12 16:51**

F.16 Lifespan

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:lifespan

Last update: **2021/06/12 16:51**

F.17 Destination Order

Return to DDS Quality of Service

To Be Completed

F.18 History

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:history

Last update: **2021/06/12 16:51**

F.19 Resource Limits

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:resource_linits

Last update: **2021/06/12 16:52**

F.20 Entity Factory

Return to DDS Quality of Service

To Be Completed

F.21 Writer Data Lifecycle

Return to DDS Quality of Service

To Be Completed

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.f_qos:writer_data_lifecycle

Last update: **2021/06/12 16:52**

F.22 Reader Data Lifecycle

Return to DDS Quality of Service

The `READER_DATA_LIFECYCLE` policy controls the behavior of the DataReader with regards to the lifecycle of the data-instances it manages, that is, the data-instances that have been received and for which the DataReader maintains some internal resources.

The `DataReader` internally maintains the samples that have not been taken by the application, subject to the constraints imposed by other QoS policies such as `HISTORY` and `RESOURCE_LIMIT`.

The `DataReader` also maintains information regarding the identity, `view_state` and `instance_state` of data-instances even after all samples have been 'taken.' This is needed to properly compute the states when future samples arrive.

Under normal circumstances the `DataReader` can only reclaim all resources for instances for which there are no writers and for which all samples have been 'taken.' The last sample the `DataReader` will have taken for that instance will have an instance_state of either `NOT_ALIVE_NO_WRITERS` or `NOT_ALIVE_DISPOSED` depending on whether the last writer that had ownership of the instance disposed it or not. The following figure provides a state chart describing the transitions possible for the `instance_state`.

In the absence of the `READER_DATA_LIFECYCLE` QoS this behavior could cause problems if the application "forgets" to 'take' those samples. The 'untaken' samples will prevent the `DataReader` from reclaiming the resources and they would remain in the `DataReader` indefinitely.

The `autopurge_nowriter_samples_delay` defines the maximum duration for which the DataReader will maintain information regarding an instance once its instance_state becomes NOT_ALIVE_NO_WRITERS. After this time elapses, the DataReader will purge all internal information regarding the instance, any untaken samples will also be lost.

The `autopurge_disposed_samples_delay` defines the maximum duration for which the DataReader will maintain samples for an instance once its instance_state becomes NOT_ALIVE_DISPOSED. After this time elapses, the DataReader will purge all samples for the instance.

Source: To Be Updated

Appendix G: Tests

Return to Reference Architecture (RA) or Return to Appendices

See: OMG: Test Information Interchange Format (TestIF)

Test Name	Description
Acceptance Testing	**Acceptance Testing** is a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users. There are various forms of acceptance testing: • User acceptance Testing • Business acceptance Testing • Alpha Testing • Beta Testing Source: https://www.tutorialspoint.com/software_testing_dictionary/acceptance_testing.htm
Black Box Testing	**Black Box Testing** is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products. Black box testing can be done in the following ways: 1. Syntax Driven Testing – This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers and language that can be represented by context free grammar. In this, the test cases are generated so that each grammar rule is used at least once. 2. Equivalence partitioning – It is often seen that many type of inputs work similarly so instead of giving all of them separately we can group them together and test only one input of each group. The idea is to partition the input domain of the system into a number of equivalence classes such that each member of class works in a similar way, i.e., if a test case in one class results in some error, other members of that class would also produce the same error. Source: https://www.geeksforgeeks.org/software-engineering-black-box-testing/
End-to-End Testing (E2E Testing)	**End-to-End Testing (E2E testing)** refers to a software testing method that involves testing an application’s workflow from beginning to end. This method basically aims to replicate real user scenarios so that the system can be validated for integration and Data Integrity. Essentially, the test goes through every operation the application can perform; to test how the application communicates with hardware, network connectivity, external dependencies, databases, and other applications. Usually, E2E testing is executed after functional and system testing is complete. Source: https://www.browserstack.com/guide/end-to-end-testing

Test Name	Description
Integration Testing	**Integration Testing** is performed to test individual components to check how they function together. In other words, it is performed to test the modules which are working fine individually and do not show bugs when integrated. It is the most common functional testing type and performed as automated testing. Generally, developers build different modules of the system/software simultaneously and don't focus on others. They perform extensive black box and white box functional verification, commonly known as unit tests, on the individual modules. Integration tests cause data and operational commands to flow between modules which means that they have to act as parts of a whole system rather than as individual components. This typically uncovers issues with UI operations, data formats, operation timing, API calls, database access, and user interface operations. Source: https://www.simform.com/functional-testing-types/
Interface Testing	**Interface Testing** is defined as a software testing type that verifies whether communication between two different software systems is done correctly. A connection that integrates two components is called an interface. This interface in the computer world could be anything like an Application Programming Interface (API), web services, etc. Testing of these connecting services or interfaces is referred to as Interface Testing. An interface is actually software that consists of sets of commands, messages, and other attributes, which enable communication between a device and a user. Source: https://www.guru99.com/interface-testing.html
Regression Testing	**Regression Testing** is re-running tests for Functional Requirements and Non-Functional Requirements to ensure that previously developed and tested software still performs after a change. If not, that would be called a **regression** as far as functionality. Changes that may require regression testing include bug fixes, software enhancements, configuration changes, and even substitution of electronic components. As regression test suites tend to grow with each found defect, test automation is frequently involved. Sometimes a change impact analysis is performed to determine an appropriate subset of tests (non-regression analysis). Source: https://en.wikipedia.org/wiki/Regression_testing
Sanity Testing	**Sanity Testing** is a subset of regression testing. Sanity testing is performed to ensure that code changes are working properly. Sanity testing is a stopgap to check whether testing for a build can proceed or not. The focus of the team during sanity testing process is to validate the functionality of the application, not to perform detailed testing. Sanity testing is generally performed on builds where the production deployment is required immediately, like a critical bug fix. Source: https://www.geeksforgeeks.org/sanity-testing-software-testing/

Test Name	Description
Smoke Testing	**Smoke Testing** is performed on a ‘new’ build given by developers to the QA team to verify if the basic functionalities are working or not. It is one of the important functional testing types. This should be the first test to be done on any new build. In smoke testing, the test cases chosen cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionality of the system is working fine. Source: https://www.simform.com/functional-testing-types/
Unit Testing	**Unit Testing** ensures that each part of the code developed in a component delivers the desired output. In unit testing, developers only look at the interface and the specification for a component. It provides documentation of code development as each unit of the code is thoroughly tested standalone before progressing to another unit. Unit tests support functional tests by exercising the code that is most likely to break. Source: https://www.simform.com/functional-testing-types/
White Box Testing	**White Box Testing** techniques analyze the internal structures of the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing, clear box testing, or structural testing. Source: https://www.geeksforgeeks.org/software-engineering-white-box-testing/

From:
https://www.omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.g_testing

Last update: **2021/06/16 11:47**

Appendix H: Acronyms

Return to Reference Architecture (RA) or Return to Appendices

Acronym	Meaning
AON	All-Or-None
API	Application Programming Interface
APP	Application
ASF	Apache Software Foundation
ASIC	Application-Specific Integrated Circuit
BFT	Byzantine Fault Tolerance
BIP	Bitcoin Improvement Proposal
CISQ	Consortium for Information & (or IT) Software Quality
CLA	Contributor License Agreements
CIL	Common Intermediate Language
CLI	Command Line Interface
CLR	Common Language Runtime
CM	Case Management
CM	Configuration Management
CoI	Community of Interest
CPU	Central Processing Unit
ĐApp	Đistributed Application
DApp	Distributed Application
DaaS	Data as a Service
dBFT	Delegated Byzantine Fault Tolerant
DBMS	Database Management Systems
DDS	Data Distribution Service
DIDO	Distributed Immutable Data Objects
DIL	Disconnected, Intermittent and Limited networks
DLT	Distributed Ledger Technology
DM	Data Model
DNS	Domain Name System
DoD	U.S. Department of Defense
DPoS	Delegated Proof of Stake
DSS	Digital Signature Standard
ECMA	European Computer Manufacturers Association
ECMAScript	European Computer Manufacturers Association Scripting Language Specification
ERC	Ethereum Request for Comment
EIP	Ethereum Improvement Proposal
EVM	Ethereum Virtual Machine
FDIC	Federal Deposit Insurance Corporation (FDIC)
FIGI	Financial Instrument Global Identifier

Acronym	Meaning
FOK	Fill-Or-Kill
GDPR	General Data Protection Regulations
GMS	Google Mobile Services
GPU	Graphical Processing Unit
gRPC	Google Remote Procedure Call
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IA	Identification and Authentication
IaaS	Infrastructure as a Service
ICO	Initial Coin Offering
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIOT	Industrial Internet of Things
IP	Intellectual Property
IP	Internet Protocol
IPFS	Interplanetary File System
IS or InfoSec	Information Security
ISO	International Organization for Standards
IT	Information Technology
ITU	International Telecommunications Union
JiT	Just-in-Time
JVM	Java Virtual Machine
K8	K8s or Kuberenetes
KYC	Know-Your-Customer
LMT	Limit Order
LSB	Linux Standard Base
MA	Mission Assurance
MKT	Market Order
MIT	Market If Touched
MIT	Massachusetts Institute Of Technology
MQ	Message Queue(MQ)
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
ODM	Ontology Definition Metamodel
OMG	Object Management Group
OpenJS	open source JavaScript
OpenMAMA	Open Middleware Agnostic Messaging API
OS	Operating System
OT	Operational Transforms
OSI	Open Systems Interconnection
OSS	Open Source Software

Acronym	Meaning
OWL	Web Ontology Language
P2P	peer-to-peer
P&P	Policy and Procedure
PaaS	Platform as a Service
PIM	Platform Independent Model
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
Protobuf	Google Protocol Buffer
PSM	Platform Specific Model
RA	Reference Architecture
RDF	Resource Description Framework
RFP	Request For Proposal
RFC	Request for Comment
RFI	Request for Information
REST	Representational State Transfer
RDBMS	Relational Database Management Systems
RESTful	RESTful API
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SCAP	Security Content Automation Protocol
SfA	Safety Assurance
SPDX	Software Package Data Exchange
SPV	Simple Payment Verification
SQuaRE	Systems and software Quality Requirements and Evaluation
STP	Stop Order
STP	Straight-through Processing (StP)
STPLMT	Stop Limit Order
SW	Software
SwA	Software Assurance
SysA	System Assurance
SysML	System Modeling Language
TCP	Transmission Control Protocol
SI	System Integrator
SIG	Special Interest Group
SDO	Standards Developing Organization
SoS	System of Systems
SPARQL	Simple Protocol and RDF Query Language
SSO	Standards Setting Organization
TODO	Talk Openly Develop Openly

Acronym	Meaning
UAF	Unified Architecture Framework
UDP	User Datagram Protocol
UID	Unique Identifier
UUID	Universal Unique Identifier
UML	Unified Modeling Language
VM	Virtual Machine
W3C	World Wide Web Consortium
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
XSLT	XSL Transformations
ZMTP	ZeroMQ Message Transport Protocol

Appendix I: Cognitive Model

Return to Reference Architecture (RA) or Return to Appendices

- **Note:** The following is based on an Engineering Governance Model developed at US Navy SPAWAR[265].

The Cognitive Model abstractly represents human cognition defined in the dictionary as[266]:

1. *The mental process of knowing, including aspects such as awareness, perception, reasoning, and judgment*
2. *That which comes to be known, as through perception, reasoning, or intuition; knowledge*

Cognition roughly maps to the Information Science and Knowledge Management DIKW (Data, Information, Knowledge and Wisdom) hierarchies[267].

Table 23 highlights the alignment of the two models to each other.

Table 23: Mapping Cognitive Aspects to DIKW Hierarchy

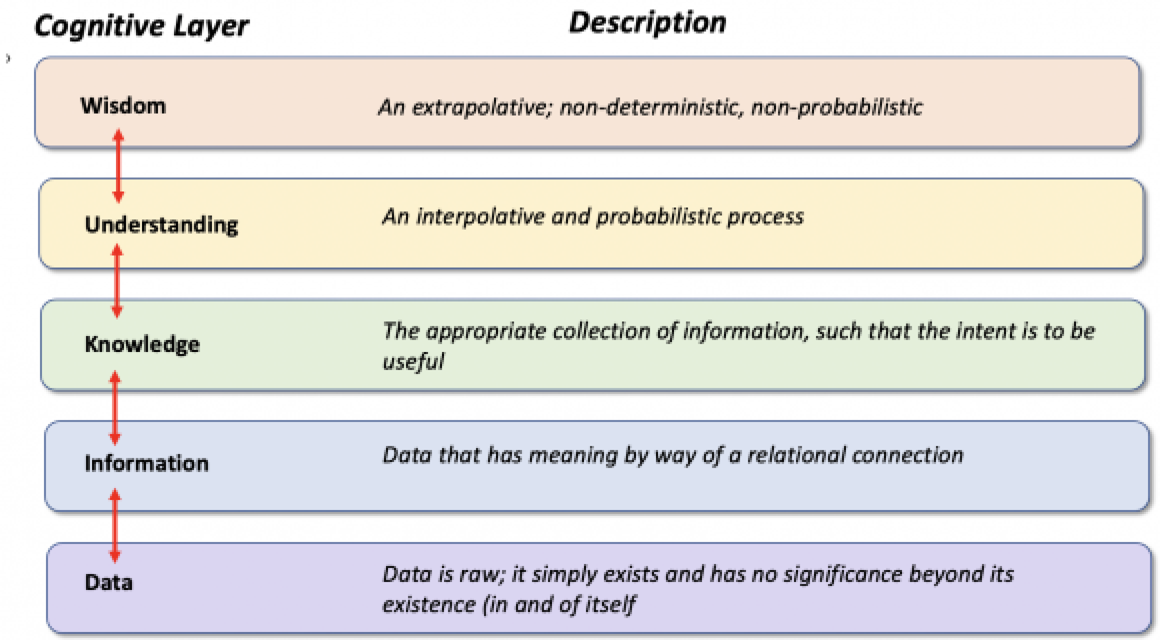
Cognitive Aspects	DIKW Hierarchy
Awareness	Data
Perception	Information
Reasoning	Knowledge
Judgment	Wisdom

In addition to the basic layers in the DIKW Hierarchy, Russell Ackoff[268] and Milan Zeleny[269] propose an additional layer between Knowledge and Wisdom. Ackoff refers to it as *Understanding*. Zeleny adds one more layer above Wisdom called *Enlightenment*. For the purposes of governance, there does seem to be a need for an *Understanding Layer* to the hierarchy. However, adding an *Enlightenment Layer* when referring to governance always seems to elicit smiles.

The result is termed the Cognitive Model instead of the original DIKW (or DIKUW) Hierarchy for a several reasons. The word hierarchy implies an order or precedence and this hierarchy always starts with data. This is a useful concept when thinking in terms of Information Science and Knowledge Management, which generally tries to organize and classify large amounts of data and extract *wisdom* or, in Zeleny's case, even *enlightenment*. In governance, the hierarchy is applicable in both directions (i.e., from *Wisdom* to *Data* and from *Data* to *Wisdom*). Another problem with the hierarchical approach is that although the relationship of data to wisdom, in some cases, is many-to-one (i.e., many pieces of data contribute to a single piece of wisdom), the reality is that the relationship is more of a network where one piece of data may ultimately be part of many pieces of wisdom.

The acronym DIKW (or DIKUW, etc.) is not very pronounceable and the term specifically captures the model as we currently understand it. Consequently, as our understanding of the model evolves, as with the acceptance of having an *Understanding Layer*, the name of the model must also change.

The Cognitive Model is depicted below in Figure 32. It has the five layers of the original DIKUW Hierarchy, and this view is from Wisdom to Data. However, the model could just as well be presented in the reverse, starting with Data and ending in Wisdom. The directionality through the model is inconsequential and reflects the higher level human cognitive process.



Cognitive Layer	Description
Wisdom	An extrapolative; non-deterministic, non-probabilistic
Understanding	An interpolative and probabilistic process
Knowledge	The appropriate collection of information, such that the intent is to be useful
Information	Data that has meaning by way of a relational connection
Data	Data is raw; it simply exists and has no significance beyond its existence (in and of itself

Figure 64: Cognitive Model

Bottom-Up Cognitive Model Example

Return to Top

The simple Bottom-Up Cognitive Model example presented in Figure 65 illustrates how bottom-up cognition applies in our lives, usually as part of analytical processes. It is bottom-up because the process described starts with Data and ends with Wisdom. At the Cognitive Data Layer, a temperature of 100° means little. Adding that the temperature is in degrees Fahrenheit provides a bit more data; however, it still has little relevance until the temperature is put in the context of a person's temperature and becomes Information. Adding that information with other information like the normal temperature for a person is 98.6° Fahrenheit starts to provide us some Knowledge of the situation. This knowledge, combined with other knowledge, allows us to understand that the person has flu. The final step is adding this knowledge with what we already know about the individual allowing a decision that the temperature is not serious and that the solution is to take two aspirin and call the doctor in the morning if symptoms persist. In reality, there is more data than information, more information than knowledge, etc.

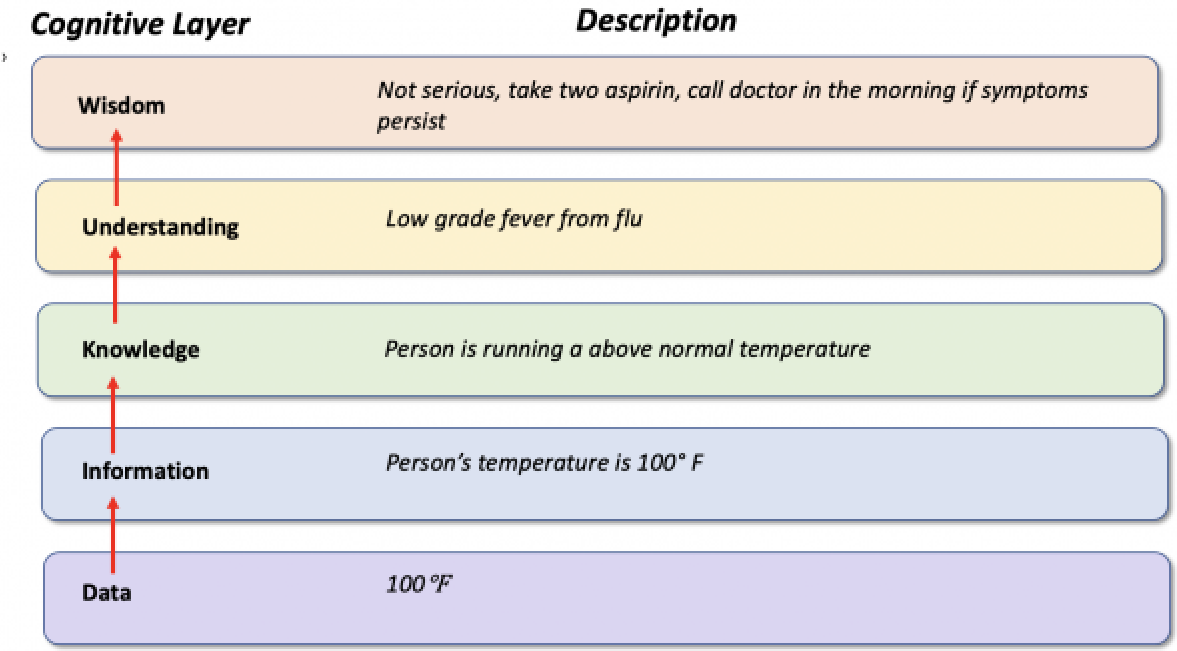
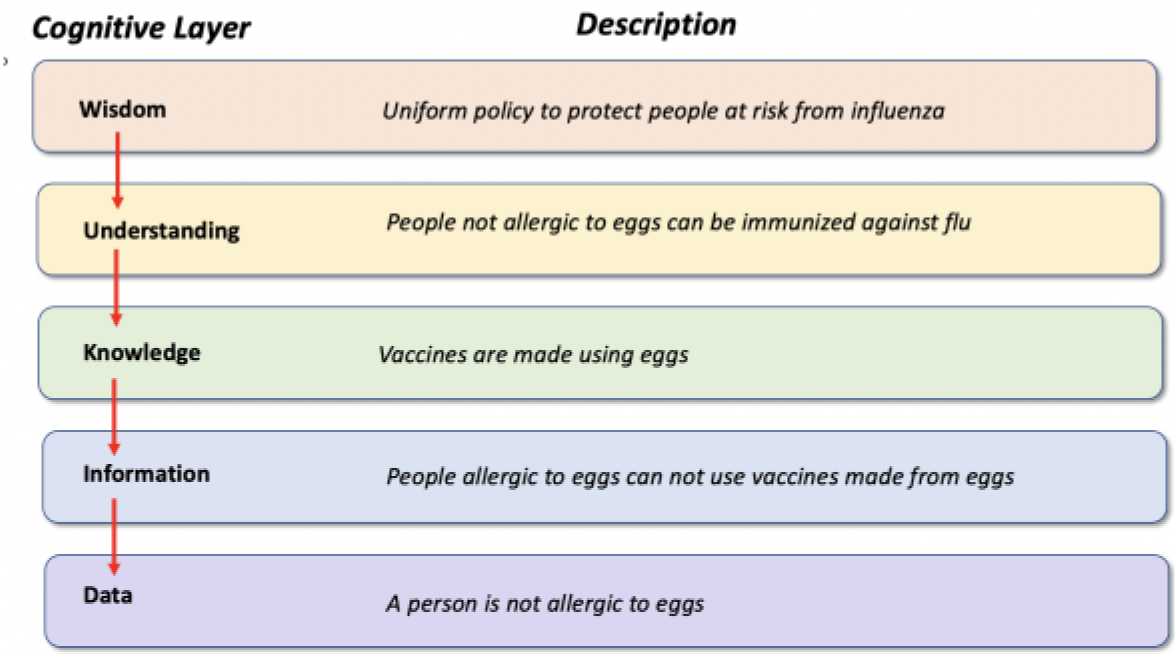

Cognitive Layer
Description
Wisdom
Not serious, take two aspirin, call doctor in the morning if symptoms persist
Understanding
Low grade fever from flu
Knowledge
Person is running a above normal temperature
Information
Person's temperature is 100° F
Data
100 °F

Figure 65: Example of Cognitive Model – Data to Wisdom

Top-Down Cognitive Model Example

Return to Top

The simple Top-Down Cognitive Model example presented in Figure 66 illustrates how top-up cognition applies in our lives, usually as part of educational or regulatory processes. It is top-down because the process described starts with Wisdom and ends with Data. At the Cognitive Wisdom Layer, there needs to be a uniform policy to protect people at risk from influenza. To support this policy (i.e., Wisdom), there are many different kinds of things to understand, such as people can be immunized against flu. As a part of the Understanding, there is Knowledge that vaccines are made from eggs. This leads to the need to disseminate Information that people who are allergic to eggs can not use the vaccine and ultimately the collection of Data (i.e., evaluation criteria) about egg allergies from people receiving the vaccine.



Cognitive Layer | Description

Wisdom | Uniform policy to protect people at risk from influenza

Understanding | People not allergic to eggs can be immunized against flu

Knowledge | Vaccines are made using eggs

Information | People allergic to eggs can not use vaccines made from eggs

Data | A person is not allergic to eggs

Figure 66: Example of Cognitive Model – Wisdom to Data

[265] Stavros, Robert W. and Albrant, Jeremiah; Engineering Governance, SPAWAR, October 9, 2007,

[266] ((American Heritage Dictionary, Accessed 10 February 2021, http://dictionary.reference.com/help/ahd4.html

[267] Nikhil Sharma, The Origin of the “Data Information Knowledge Wisdom” Hierarchy, 2008 https://www.researchgate.net/publication/292335202_The_Origin_of_Data_Information_Knowledge_Wisdom_DIKW_Hierarchy

[268] Ackoff, Russell; 1981, pp. 15-16

[269] Zeleny, M., ed. (1981) Autpoiesis: A theory of living organization, vol.3, New York, New York, American Elsevier

Appendix J: Governance Model

Return to Reference Architecture (RA) or Return to Appendices

Fundamental Governance Model

- **Note:** The following is based on an Engineering Governance Model developed at US Navy SPAWAR[36].

Governance is not a synonym for government or for regulations; rather, governance is the *process* governments use to *interpret* and *use* regulations.

> *Governance is that separate process or certain part of management or leadership processes that make decisions that define expectations, grant power, or verify performance. Frequently a government is established to administer these processes and systems.* [37]

This definition offers three aspects as to what comprises governance:

- Making decisions that define expectations
- Granting power
- Verifying performance

The first aspect of governance conveys regulation, the second aspect conveys execution, and the third aspect conveys compliance, as represented in the following model.

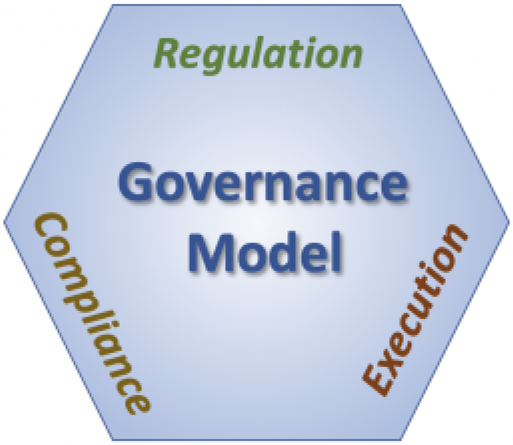


Figure 7: General Governance Model

Good governance is comprised of all three components in equal and sometimes opposing aspects. It is meaningless to have regulation without execution or execution without compliance. In other words, regulation indicates what needs to be done, execution is actually doing it, and compliance is making sure it is done correctly.

Regulation

Return to Top

Regulations are formal, codified authoritative rules. They are adopted by a public regulatory agency and are usually interpretations of statutes passed by a legislative body.

A regulation as a legal term is a rule created by an administration or administrative agency or body that interprets the statutes setting out the agency's purpose and powers, or the circumstances of applying the statute. A regulation is a form of secondary legislation which is used to implement a primary piece of legislation appropriately, or to take account of particular circumstances or factors emerging during the gradual implementation of, or during the period of, a primary piece of legislation.[38)]

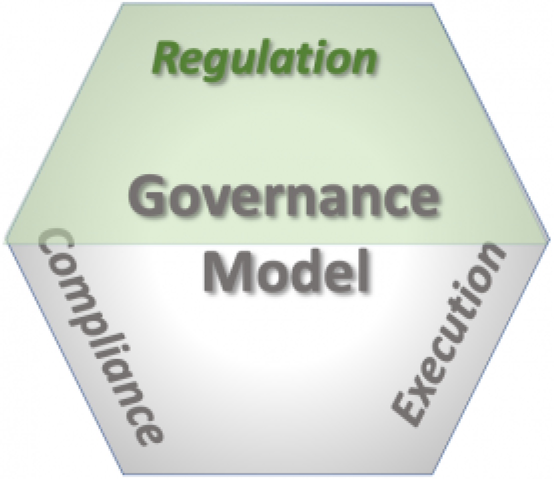


Figure 8: Regulation Aspect of Governance Model

An example of the regulatory aspect of governance is a body that creates statutes such as the U.S. Congress or a state legislature, an agency that creates or enforces statutes such as the U.S. Internal Revenue Service (IRS), or civilian or commercial groups that create and promote standards such as the Object Management Group (OMG).

Compliance

Return to Top

Compliance ensures that regulations are met through observation, measurement or testing. Good governance cleanly and effectively separates the responsibility for creating regulations from the enforcement of these regulations. This does not mean that regulations can be developed in a vacuum; they must be written to be enforceable using objective measures of compliance. Therefore, the line between regulation and compliance is not fixed and rigid but needs to be negotiated with validation of regulations from the compliance aspect.

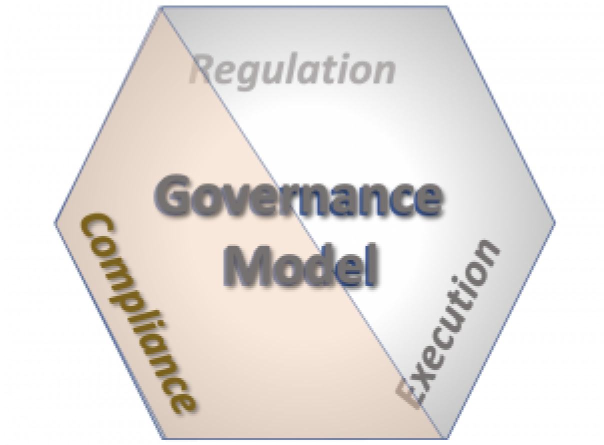


Figure 9: Compliance Aspect of Governance Model

Examples of the Compliance Aspect of Governance is the auditing functionality of the IRS and the independent verification and validation (IV&V) functionality within the DoD.

Execution

Return to Top

Execution is the aspect of governance charged with actually fulfilling formal, codified authoritative rules specified by regulation to those specifications provided by compliance. The responsibility for executing the regulation rarely, if ever, falls on the legislative body or those responsible for enforcing compliance with the regulation. Without execution, the other aspects of Governance are meaningless. Consequently, any discussion of governance must include the Execution Aspect.

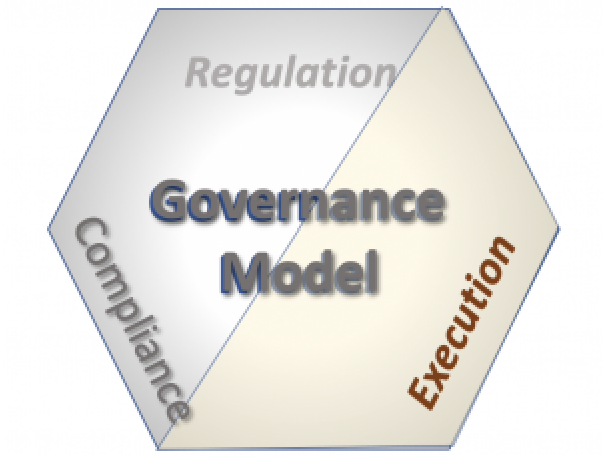


Figure 10: Execution Aspect of Governance Model

Examples of the Execution Aspect are the individuals that file their tax forms with the IRS and the personnel who actually create the functionality needed by a DoD Program, Project, or Initiative.

[36)]

Stavros, Robert W. and Albrant, Jeremiah; Engineering Governance, SPAWAR, October 9, 2007,

[37] Adapted from Wikipedia: Governance; accessed 9 July 2007

[38] Adapted from Wikipedia: Regulation, accessed 9 July 2007 https://en.wikipedia.org/wiki/Regulation]]