Accelerating MBSE Impacts Across the Enterprise: Model-Based S*Patterns

William D. Schindel[1], Stephen A. Lewis[1], Jason J. Sherey[1], Saumya K. Sanyal[2]

[1]ICTT System Sciences [2]K2 Firm, LLC

schindel@ictt.com, lewis@ictt.com, sherey@ictt.com, sksanyal@k2firm.com

Abstract. Model-Based Systems Engineering (MBSE) methods can directly address "silos" problems. This paper reports on work by the INCOSE MBSE Initiative Patterns Challenge Team, focusing on Pattern-Based Systems Engineering (PBSE) using model-based system patterns based on the S*Metamodel, speeding and improving multiple SE processes.

Distinctive are (1) the configurable, model-based nature of the patterns (not all historical patterns work has been model-based), (2) the technical scope of the models, encompassing requirements, design, failure mode, verification, other aspects, (3) the system scope of the models, encompassing whole systems, configurable product lines, and platforms, not just libraries of components, (4) the diverse and integrating cross-enterprise domains of the patterns, encompassing products, innovation processes, manufacturing, packaging / distribution, and other domains, and (5) the ability to enable a variety of COTS modeling languages and tools, PLM, and other enterprise information systems to integrate support of management and application of S*Patterns across enterprises.

Introduction

Business Challenges and Opportunities

Enterprise-level economic and competitive pressures, along with human nature, can drive managers within product manufacturing or other complex enterprises into understandably defensive postures and responses. These may locally defend their departmental functions, but globally sub-optimize the performance of the enterprise—risking the viability of the organization and the well-being of all within it. Major departments (Research, Development, Engineering, Marketing, Finance, Accounting, Production, etc.) are struggling to maintain performance, cost, quality, and schedule standards set by others, with shrinking budgets for capital, R&D, and overhead budgets. Information Technology is simultaneously a cost center, a key enabler, and a major gating factor for organizational performance. Human Resources departments may find they are only able to manage compliance and crises, but not be responsive to requests for proactive resource planning and development.

Competitive pressures of markets for products and services, as well as investor market forces, are driving the need to shorten innovation cycle time from concept to production, accompanied by demand to deliver products with increased feature sets at reduced purchase and operational costs. Cars are expected to be safer, reduce driver load, be more comfortable, and contain sophisticated electronics, all at reduced fuel consumption and maintenance cost. Cell phones are expected to have greater storage and display capacity, digital connectivity, and voice quality with significantly extended battery life. Consumable goods must concurrently be effective and meet increasingly complicated set of safety and environmental regulation.

These challenges are not limited to the introductory cycle of innovations, but can also be found throughout the subsequent life cycles of the new or innovated systems. Also common across all of these examples are a backdrop of growing complexity and the constant drumbeat of cheaper, cheaper, and cheaper.

Companies, managers, and industries that deal with these pressures well not only survive but thrive. Part of the art and science of this success is concerned with addressing “enterprise level” coordination that must occur across different functional areas to achieve an emergent result for customers, shareholders, and other stakeholders. While teamwork and culture are essential ingredients of that success, the fundamental nature of cross-organizational interactions is itself a growing systems challenge that cannot be entirely overcome with culture and good will, as expectations, complexity, and speed continue to increase.

An explicit enterprise level systems model can help management and staff because:

- Individual managers and staff don’t otherwise have broad enough experience and internalized grasp of underlying interactions to know how to optimize cross-departmental interactions to benefit the broader enterprise,
- Individual managers and staff may not be measured or rewarded on their contribution to enterprise level results in a way that can be tangibly identified as being their contribution,
- It can be difficult to understand and manage cost/benefit or investment/return for a function that is implemented in one department and which delivers benefits in another.

A variety of tools and methods have been brought to this party, and their success is observed and studied as best practices are codified and spread virally crossing industry and functional boundaries. Enterprise information systems, first invented to solve complex scheduling problems in manufacturing (e.g., MRP, ERP, etc.), are now being extended and used to manage financial and human resources. Lean and Six Sigma Tools have been adapted to every functional silo. Collaborative information technology and social media access have outstripped their ability for governance. With that said, in our respective client practices we find that localized private (spreadsheet or other) data is frequently still viewed (based on behavior we observe, across citizen service, defense, aerospace, automotive, health care, advanced manufacturing, energy, telecom) as the most useful information systems and legacy data management tool, while simultaneously symbolic of our clients’ biggest challenges.

Global level progress does occur. Project Management, Capability Maturity, and Risk Management in the late 1980s were thought by some to be so burdensome that defense systems engineers conversant in these techniques were relatively unemployable in the commercial sector. Today PMI, CMMI, and Risk Management are common valued skill sets for project managers in home construction, hospitality, education, and consumer products. Systems engineering methods and tools formerly reserved for society’s most ambitious undertaking are finding their way into use to serve multi-domain issues across Products, Manufacturing Processes, Supply Chain and Distribution Systems. While credit for some of this evolution ought to be given to the improved maturity and accessibility of those methods and tools, it is fair to say that it is also being driven by growing complexity in these formerly “simpler” domains—whether the methods and tools are fully ready yet or not.

This paper exemplifies the MBSE shift of emphasis from phenotype to genotype, from atoms to bits, from making things to printing things, from bricks and mortar to intellectual property, and from expertise of the individual to shared team knowledge—including at the enterprise level. The competitive game is moving to the Model Based Economy. What better toolkit could we ask for than MBSE? How can we harness its promise across the Enterprise? Where is additional progress needed?

*Background on MBSE/PBSE and the S*Metamodel*

The Patterns Challenge Team of the INCOSE MBSE Initiative (INCOSE Patterns Team 2014) was formed in 2013 to pursue the practical use and awareness of system patterns of a particular type, called S*Patterns, which are described as follows:

1. **S*Models** are MBSE models that are based on the S*Metamodel. (The Metamodel provides an underlying framework that defines the semantic meaning of models conforming to it.) The S*Metamodel's explicit semantics include some key system concepts that are long-established in science and engineering, but not always found to be so explicit in contemporary models (Schindel 2005a,b, 2011a,b,c, 2013a). Figure 1 is a summary extract of some of the most important aspects of the underlying S*Metamodel, described in greater detail in those references.

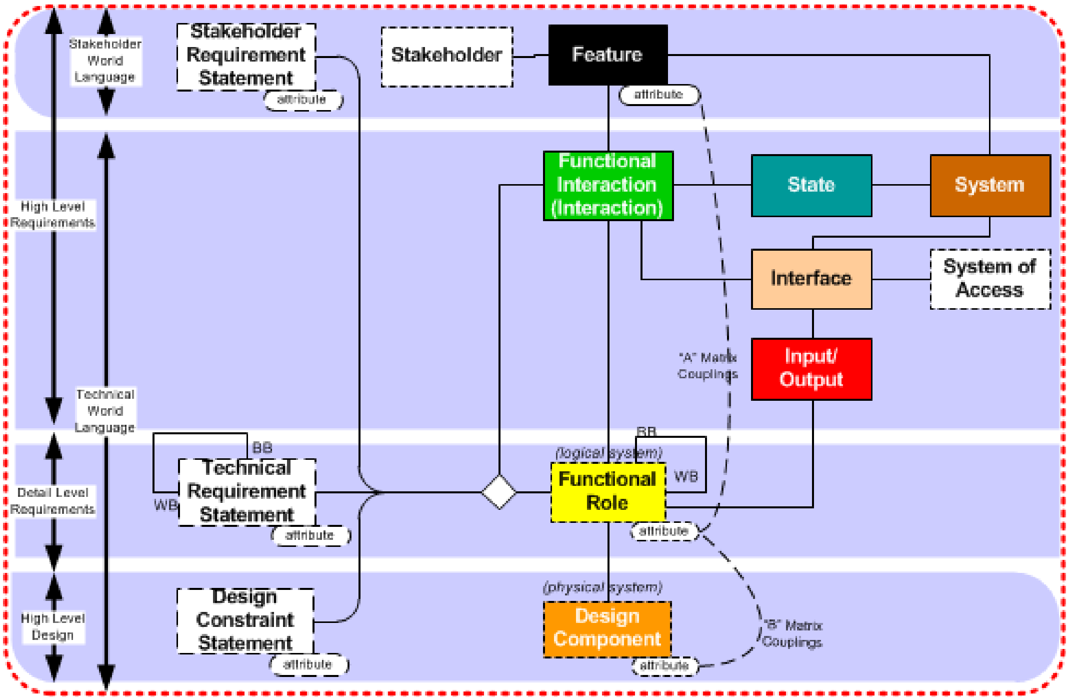


Figure 1: Summary of Some Key Elements from the S*Metamodel

2. **S*Patterns** are configurable, re-usable S*Models. (Not all historical system pattern work was based on the use of models, but S*Patterns are.) An S*Pattern may be thought of as a model of a family of systems, a platform, or a product line, or as an architectural framework. As shown in Figure 2, once an S*Pattern has been created for a given enterprise, product line, or other domain, it may be used during a delivery project to rapidly create a high-grade S*Model, typically an order of magnitude faster than by creating a new model, and configured for the specific needs at hand (Schindel 2005a, 2011c, 2012a,b, 2014, Schindel and Peterson 2013, Schindel and Smith 2002, Bradley, Hughes, Schindel 2010, Cloutier 2008, Alexander 1977, Gamma et al 1995, ISO 42010 2011).

3. S*Models and S*Patterns are independent of any specific **modeling language**, and are typically expressed using any of a variety of the popular standard or third-party contemporary modeling languages, once a mapping is provided. (For example, in this paper some of the S*Models are expressed in SysML language.) This has the impact of strengthening the semantics of existing modeling languages in areas necessary to support key historical practices of engineering and science (Schindel 2005b, 2010, 2014).

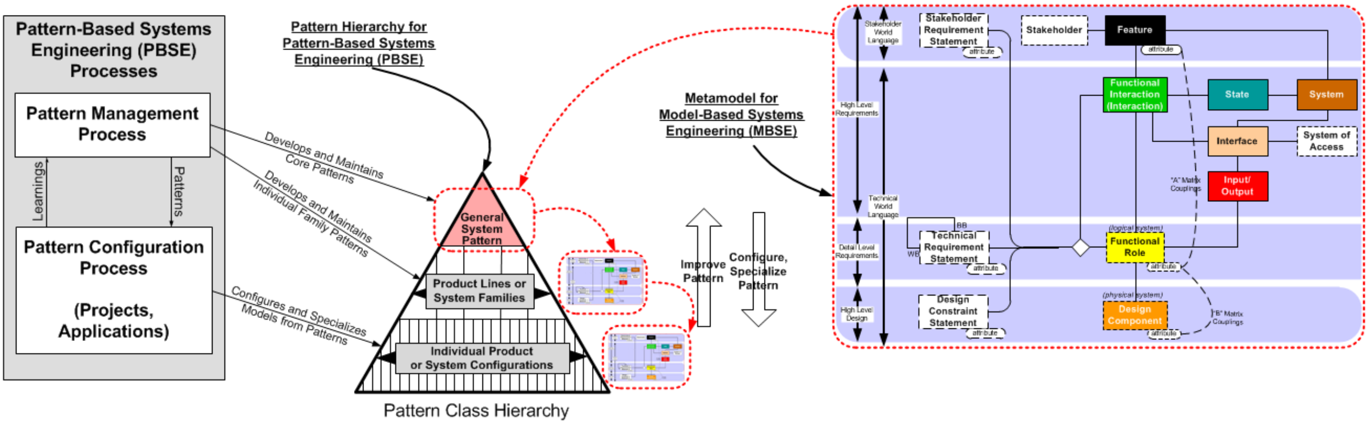

Pattern-Based Systems Engineering (PBSE) Processes
Pattern Management Process
Learnings
Patterns
Pattern Configuration Process
(Projects, Applications)
Pattern Hierarchy for Pattern-Based Systems Engineering (PBSE)
Develops and Maintains Core Patterns
Develops and Maintains Individual Family Patterns
Configures and Specializes Models from Patterns
General System Pattern
Product Lines or System Families
Individual Product or System Configurations
Pattern Class Hierarchy
Improve Pattern
Configure, Specialize Pattern
Metamodel for Model-Based Systems Engineering (MBSE)
Stakeholder World Language
High Level Requirements
Technical World Language
Detail Level Requirements
High Level Design
Stakeholder Requirement Statement
attribute
Stakeholder
Feature
attribute
Functional Interaction (Interaction)
State
System
Interface
System of Access
"A" Matrix Couplings
Input/ Output
BB
WB
Technical Requirement Statement
attribute
(logical system)
Functional Role
attribute
Design Constraint Statement
attribute
(physical system)
Design Component
attribute
"B" Matrix Couplings

Figure 2: S*Patterns Are Configured to Generate S*Models

4. S*Models and S*Patterns are independent of any specific **software tool or information system**, and may be stored in and managed by a variety of popular third-party COTS modeling and engineering tools and information systems, once a mapping is provided. (For example, this paper illustrates S*Models and S*Patterns in several of the third party COTS system modeling tool, requirements database, and PLM systems that have been used in various domains.) This has the impact of increasing the value of existing COTS system modeling tools, requirements databases, and PLM systems already in use.

5. The processes of Systems Engineering consume and produce **information**. However, there is a long tradition of extensive descriptions of the **process and procedure** of Systems Engineering. Compared to the amount of ink and effort traditionally spent to describe SE process and procedure, the amount spent to describe information (which passes through those processes and procedures) is usually orders of magnitude less. Compare this to the amount of description of the underlying relationships of physics, chemistry, or electronics, versus the description of related engineering procedures. This imbalance was somewhat understandable in the day in which Systems Engineering information was in the form of prose, for which underlying "theory" is limited, but in the current day in which that information can be based on explicit models—the language of science and mathematics—we suggest a shift in this balance is in order. Figure 3 illustrates the model of both the SE process and the information passing through it, and the idea that the SE process should be primarily performed to drive trajectories in configuration space (Schindel 2015a,b).

6. The processes of MBSE as typically practiced today (Estafan 2008), as well as more traditional Systems Engineering (ISO 15288, INCOSE SE Handbook 2014) are most often presented, conceived, or practiced as if each engineering project is "starting from

scratch" to "green field" conceptualize a new system of a sort never before conceived. Much procedural guidance is offered as to the discovery, study, synthesis, and analysis of stakeholders, requirements, allocations, and architectures, trade-spaces, risks and failure modes, etc., in a context that might lead one to believe the system of interest is being studied for the first time. Although nothing about this good guidance is inappropriate in principle to engineering the next generation of established domain systems, there is a relatively low balance of guidance on the formal inclusion of what we already know with discovery of what is new. The "up stroke and down stroke" of Figure 2, deal with the relationship between managing formal model-based patterns of what is already known (similar to the physical sciences), configuration of that information to specific projects, and the interplay of the two. Recent progress with Product Line Engineering illustrates a start on progress to rebalance this situation (ISO/IEC 26550 2013).

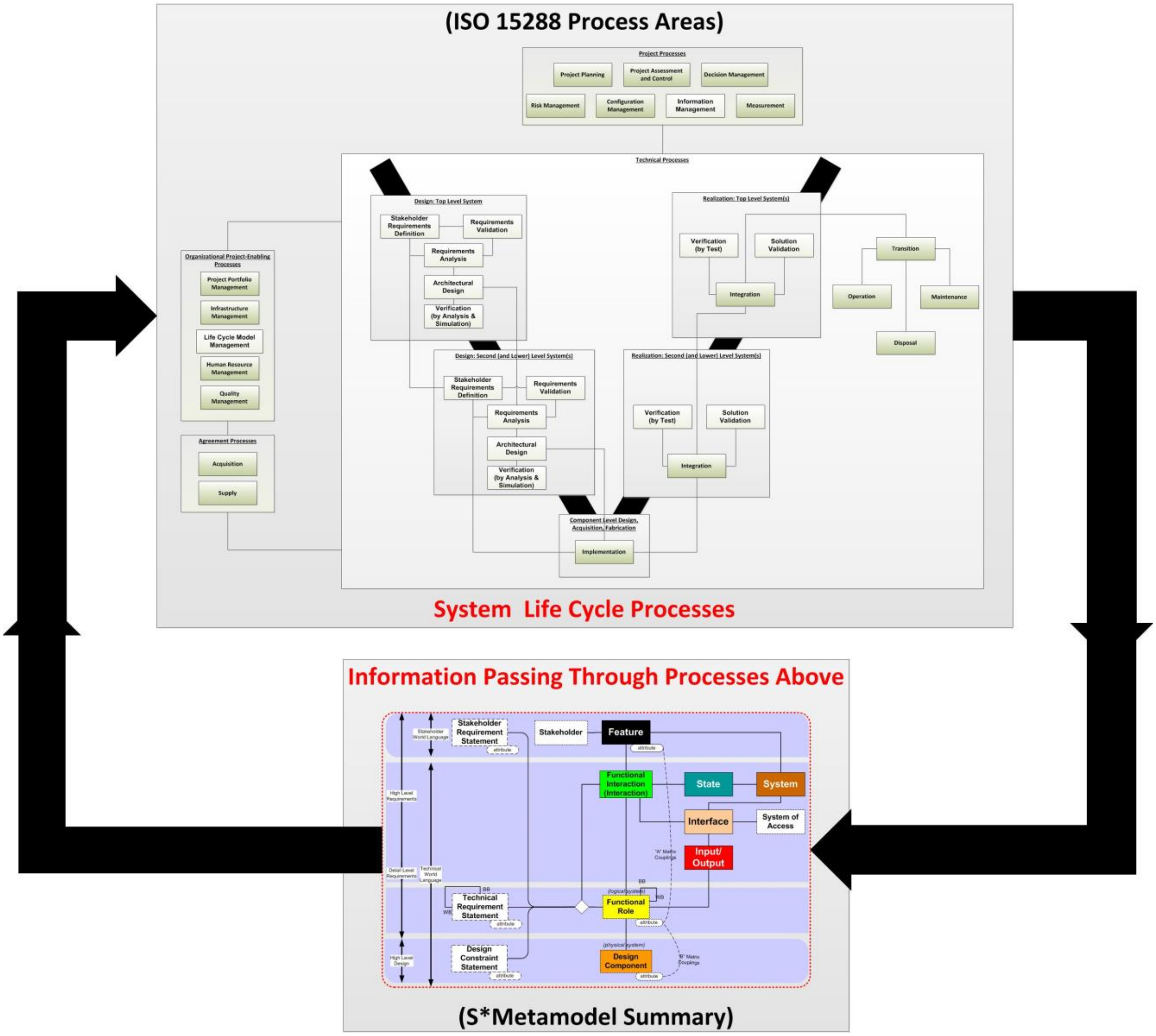


Figure 3: The MBSE Process Consumes and Generates S*Models

The Patterns Challenge Team has practiced use of S*Patterns to describe autonomous ground vehicles, automated safety critical system test, optimization of design review assignment, and (in this paper, below), cross-functional enterprise dependencies in product manufacturing businesses (Peterson et al 2015; Cook et al 2015; Nolan et al 2015).

Integrating S*Patterns, at Enterprise and Lower Levels

Agricultural silos (Figure 4) are designed to minimize unwanted external interactions that are harmful to stored silage. The "silos" metaphor is an infamous description invoked to describe all-too-frequent organizational pathologies of a certain type—those in which lack of coordination, cooperation, teamwork, or alignment across parts of the organization rob the enterprise of optimum performance. This can be unfair or viewed as an attack on the hard-working managers and value-generating staff in functional areas, when it is in fact an emergent aspect of the overall enterprise system. Dealing with this situation at a systems level, beginning by representing the system more transparently, provides a more positive way to engage those playing roles in these systems in future innovation.

Figure 4: Silos—Good for Farms, Bad for Organizations

In the enterprise case, the systems engineer's interpretation focuses on the interactions (or lack of them) between the functional areas of the enterprise, along with external actors. The enterprise is a system, and a system is a collection of interacting parts. Based on those internal interactions (exchanges of information, mass flows, energy, forces), an overall enterprise behavior emerges, as "seen" by the external "actors" (for example, customers) through the external interactions with the "black box" enterprise, as in Figure 5.

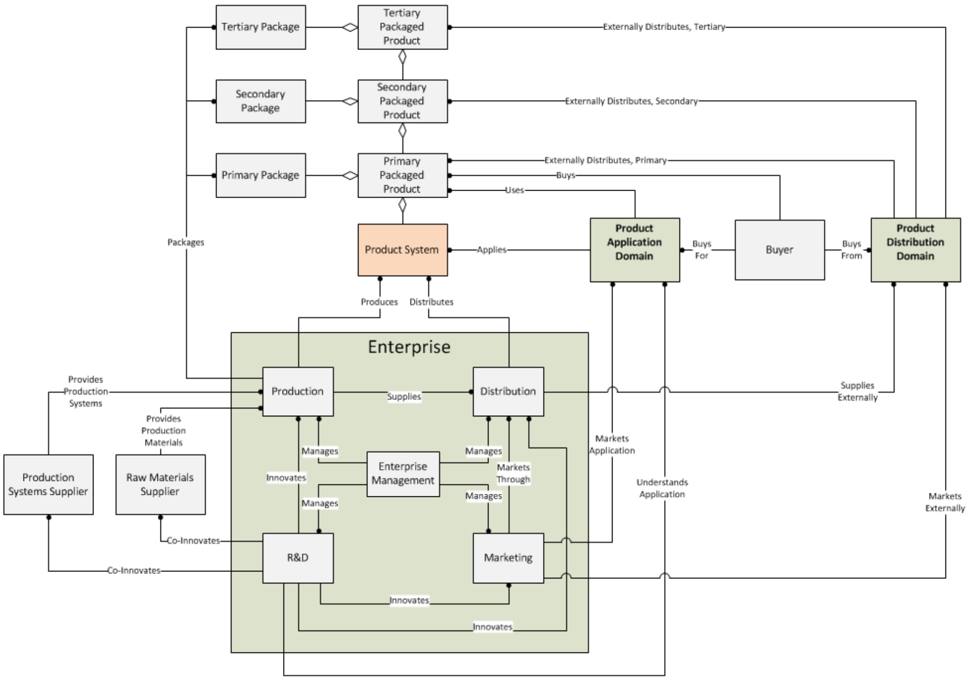


Figure 5: The Enterprise As System Embedded In Its Domain—An Example

The interactions of this perspective are a basic fact of nature about any organization, whether high-performing or not, whether healthy culture of not, and across all business and institutional models and domains. Just as we must not ignore the emergent characteristics of a designed product or a system-of-systems, likewise we ignore this aspect of organizations at our peril.

Instead of overlooking these challenges, we describe here how they can be embraced as sources of competitive advantage, built directly into the formalisms and information systems that help define the enterprise and its local and global practices. This begins by adopting an explicit model that focuses attention on the important interactions across organizational functions, using the Enterprise System Pattern.

The Enterprise System Pattern

For a given enterprise, the Enterprise System Pattern is an S*Pattern that can be configured for individual enterprise-level projects or other endeavours. This pattern is created once for the enterprise, but thereafter updated as learning occurs. The "system of interest" for this S*Pattern is the enterprise, illustrated by the Top Level System in the Vee diagram of Figure 3, and the Enterprise Domain Model of Figure 5.

Like all S*Patterns, the Enterprise Pattern includes all the S*Metamodel aspects, for which Figure 1 provides a summary. For this paper, two aspects of particular interest are:

1. **<u>Functional Interactions (a fundamental part of all S*Models) that span multiple subsystems of the Enterprise</u>**, or other Domains, visible in Figure 5. It is these "cross functional" interactions (or their absence) that are the source of "Silo Pathologies" of the Enterprise. The solution is to understand and manage the interaction as a whole, and this begins with its representation in a system model at the Enterprise level. An example benefit is illustrated later below.

2. **<u>The Enterprise Management subsystem</u>** is shown in the logical architecture of Figure 5. It is the "tip of the iceberg" of the Management System Pattern (aka the Embedded Intelligence (EI) Pattern). Figure 6 illustrates the hierarchy of (human and automated) Management Systems. Some of these appear again playing management and controls roles in enterprise subsystems of Figure 5, in later sections below.

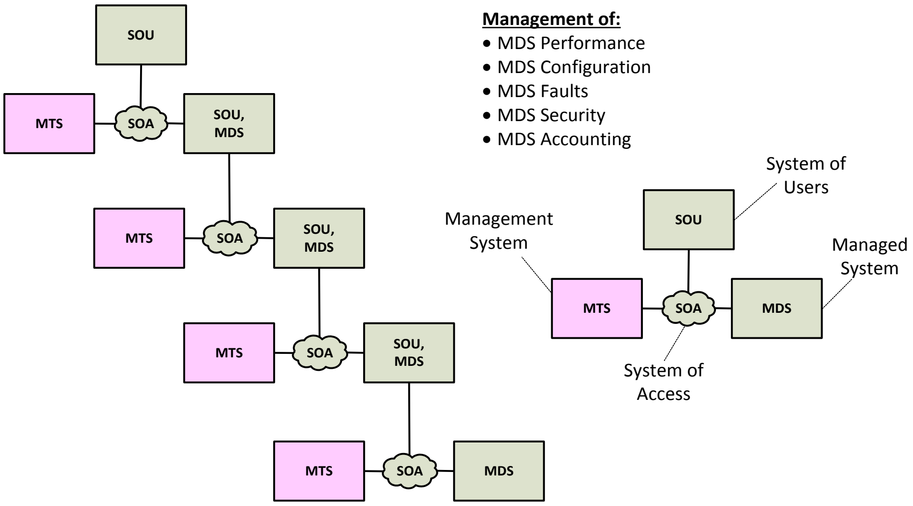


Figure 6: The EI Hierarchy of the Embedded Intelligence Pattern

A key emphasis of this paper is the importance of explicitly modelling and managing the Enterprise level system, for successful enterprise projects. This can be very effectively performed by the PBSE approach, summarized in Figure 2:

1. Pattern Management Process: Creating and improving the configurable, re-usable Enterprise S*Pattern in an appropriate modelling tool. S*Patterns may be so managed in a number of popular third party modelling tools, some of which are illustrated in this paper. This part could be viewed as establishing the S*Model minimum for the content of an Enterprise architectural framework, as in (ISO 42010 2011).
2. Pattern Configuration Process: For each major enterprise project or endeavour, configuring that pattern as an S*Model of that project. This does not necessarily require a full modelling tool, and such project-specific configured models can be managed in a PLM system, over the life cycle of the project (or product), as in Figure 7. S*Models may be so managed in a number of popular PLM, modelling tool, or other information systems.

The connection from (1) to (2) above is the S*Pattern Configuration Agent, which can be attached to a number of different third party modelling or PLM systems. Part of the Pattern Configuration Process is illustrated in Figure 12.

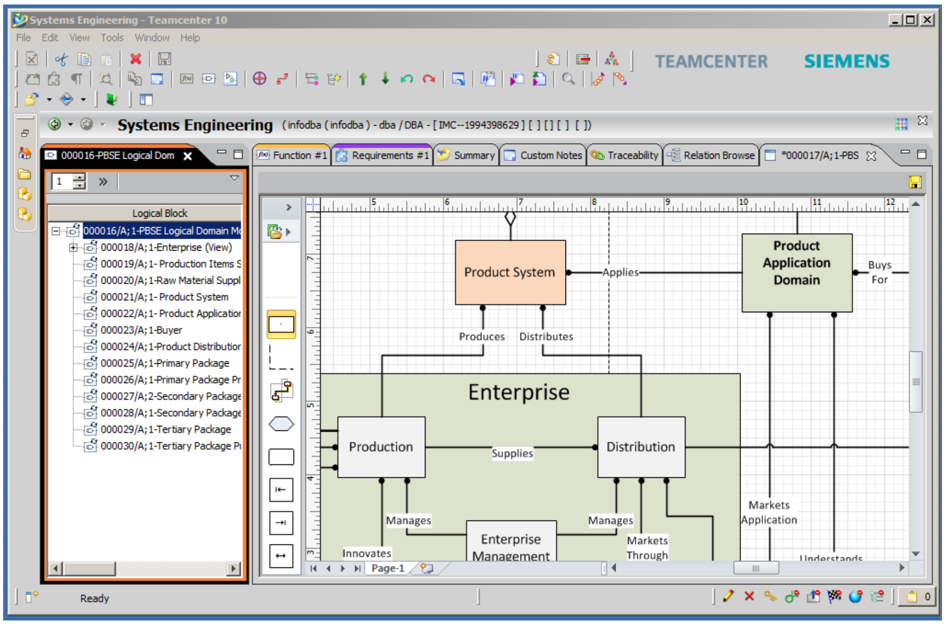

Systems Engineering - Teamcenter 10
File Edit View Tools Window Help
TEAMCENTER
SIEMENS
Systems Engineering (infodba (infodba) - dba / DBA - [IMC--1994398629] [ ] [] [ ] [ ])
000016-PBSE Logical Dom
Function #1
Requirements #1
Summary
Custom Notes
Traceability
Relation Browse
*000017/A;1-PBS
Logical Block
000016/A;1-PBSE Logical Domain Mo
000018/A;1-Enterprise (View)
000019/A;1- Production Items S
000020/A;1-Raw Material Suppl
000021/A;1- Product System
000022/A;1- Product Applicatior
000023/A;1-Buyer
000024/A;1-Product Distributior
000025/A;1-Primary Package
000026/A;1-Primary Package Pr
000027/A;2-Secondary Package
000028/A;1-Secondary Package
000029/A;1-Tertiary Package
000030/A;1-Tertiary Package Pr
Product System
Applies
Product Application Domain
Buys For
Produces
Distributes
Enterprise
Production
Supplies
Distribution
Manages
Manages
Markets Application
Enterprise Management
Markets Through
Innovates
Understands
Page-1
Ready

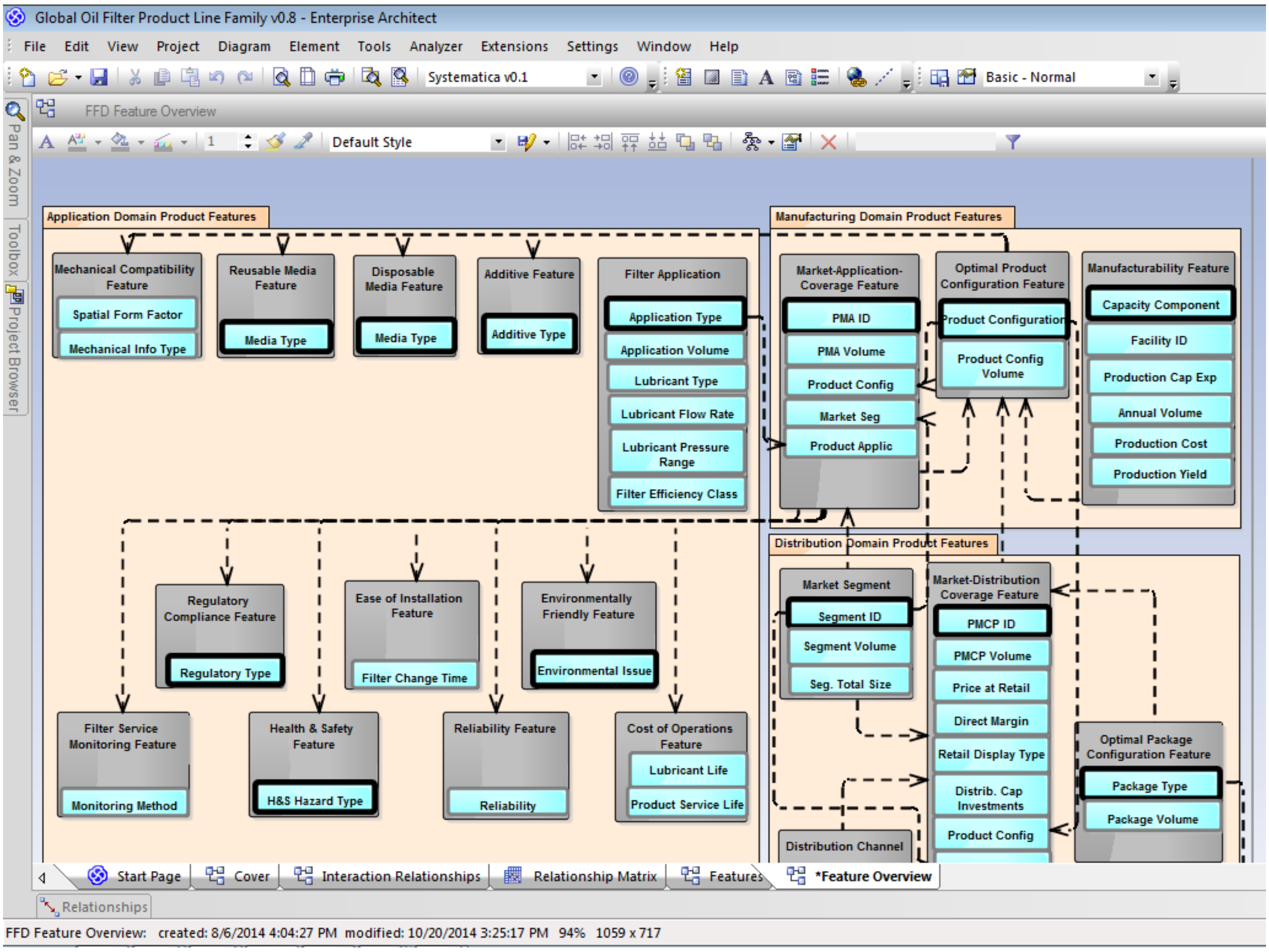
Figure 7: PLM System: A Natural Model Life Cycle Repository for an Enterprise Pattern That Has Been Configured for a Project

The Product Application Domain Pattern

The Product Application Domain Pattern, another S*Pattern, describes the enterprise's product (or platform, product line, family) in service in its intended application domain. This pattern also includes all the S*Metamodel aspects, including those summarized by Figure 1. Like the Enterprise Pattern described above, the Product Pattern is managed in some modelling environment, separately configured for each product configuration or project, and those configured S*Models are suitable to be managed in a PLM, modelling tool, or similar information system.

For some classes of products, it is most efficient for the scope of this pattern to include the product's packaging—consumer products and pharmaceuticals are typical examples. For other product types, the packaging aspects are modelled as part of the Distribution Domain.
A subset of the views typical of S*Models are illustrated below for an example family of manufactured products—the Oil Filter Product Line. The views shown in this section illustrate the use of OMG SysML modelling language and tools, all of which can be readily mapped to the S*Metamodel. There are equivalents if other modelling languages and tools are used:

1. A **<u>Stakeholder Feature Model</u>** describes the set of configurable features available in the product line. This part of the S*Pattern marks the point at which configuration for specific products will occur. Its stakeholder attribute set establishes the trade space for the system family. See Figure 8.



FFD Feature Overview: created: 8/6/2014 4:04:27 PM modified: 10/20/2014 3:25:17 PM 94% 1059 x 717

Figure 8: Stakeholder Features Overview

2. A **<u>Domain Model</u>** describes the external domain environment that the subject system (Packaged Product Oil Filter, in this case) will encounter and physically interact with, over its life cycle, ultimately traceable to all system functional requirements and stakeholder features. See Figure 9.
3. A **<u>Logical Architecture Model</u>** describes the partitioning of the system in the logical subsystems, short of their allocation to the physical architecture, also part of the configurable S*Pattern. See Figure 10.

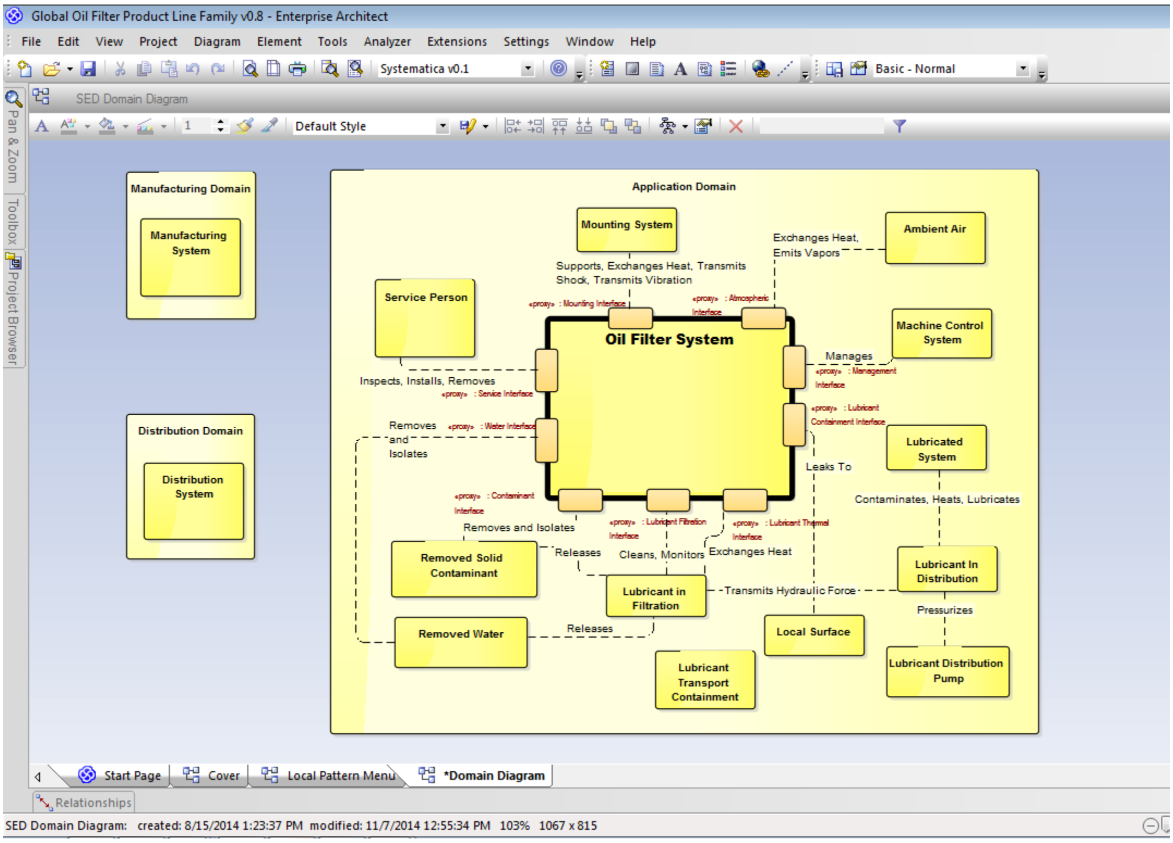

Global Oil Filter Product Line Family v0.8 - Enterprise Architect
File Edit View Project Diagram Element Tools Analyzer Extensions Settings Window Help
Systematica v0.1
Basic - Normal
SED Domain Diagram
Default Style
Pan & Zoom
Toolbox
Project Browser
Manufacturing Domain
Manufacturing System
Distribution Domain
Distribution System
Application Domain
Mounting System
Ambient Air
Exchanges Heat, Emits Vapors
Supports, Exchanges Heat, Transmits Shock, Transmits Vibration
Service Person
«proxy» : Mounting Interface
«proxy» : Atmospheric Interface
Oil Filter System
Machine Control System
Manages
«proxy» : Management Interface
Inspects, Installs, Removes
«proxy» : Service Interface
«proxy» : Lubricant Containment Interface
Removes and Isolates
«proxy» : Water Interface
Lubricated System
Leaks To
Contaminates, Heats, Lubricates
«proxy» : Contaminant Interface
Removes and Isolates
«proxy» : Lubricant Filtration Interface
«proxy» : Lubricant Thermal Interface
Removed Solid Contaminant
Releases
Cleans, Monitors
Exchanges Heat
Lubricant In Distribution
Lubricant in Filtration
Transmits Hydraulic Force
Pressurizes
Removed Water
Releases
Local Surface
Lubricant Transport Containment
Lubricant Distribution Pump
Start Page
Cover
Local Pattern Menu
*Domain Diagram
Relationships
SED Domain Diagram: created: 8/15/2014 1:23:37 PM modified: 11/7/2014 12:55:34 PM 103% 1067 x 815

Figure 9: Domain Model

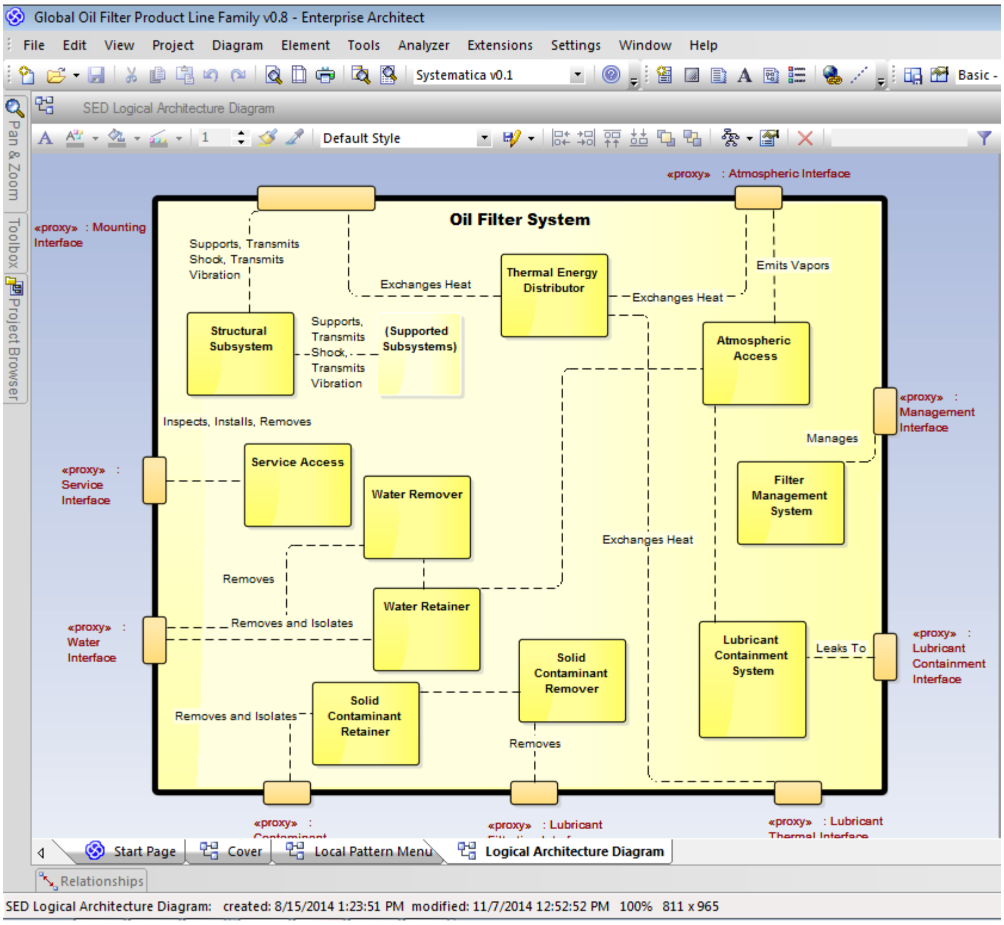
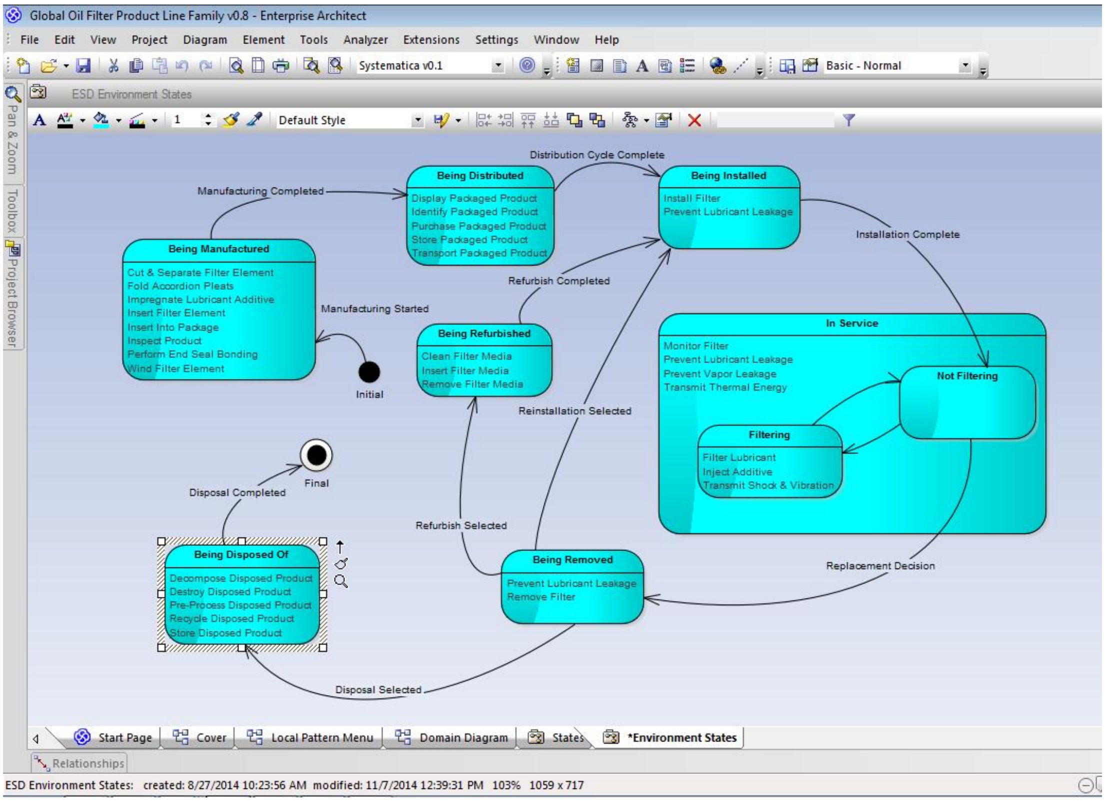

Global Oil Filter Product Line Family v0.8 - Enterprise Architect
File Edit View Project Diagram Element Tools Analyzer Extensions Settings Window Help
Systematica v0.1
Basic -
SED Logical Architecture Diagram
Default Style
Pan & Zoom
Toolbox
Project Browser
«proxy» : Atmospheric Interface
«proxy» : Mounting Interface
Oil Filter System
Supports, Transmits Shock, Transmits Vibration
Exchanges Heat
Thermal Energy Distributor
Exchanges Heat
Emits Vapors
Structural Subsystem
Supports, Transmits Shock, Transmits Vibration
(Supported Subsystems)
Atmospheric Access
«proxy» : Management Interface
Inspects, Installs, Removes
Manages
«proxy» : Service Interface
Service Access
Water Remover
Filter Management System
Exchanges Heat
Removes
«proxy» : Water Interface
Removes and Isolates
Water Retainer
Lubricant Containment System
Leaks To
«proxy» : Lubricant Containment Interface
Solid Contaminant Remover
Removes and Isolates
Solid Contaminant Retainer
Removes
«proxy» : Contaminant
«proxy» : Lubricant
«proxy» : Lubricant Thermal Interface
Start Page
Cover
Local Pattern Menu
Logical Architecture Diagram
Relationships
SED Logical Architecture Diagram: created: 8/15/2014 1:23:51 PM modified: 11/7/2014 12:52:52 PM 100% 811 x 965

Figure 10: Logical Architecture Model

4. A **State Model** describes the temporal framework of system states, modes, or situations, including what system Interactions are expected to occur during each such state. See Figure 11.



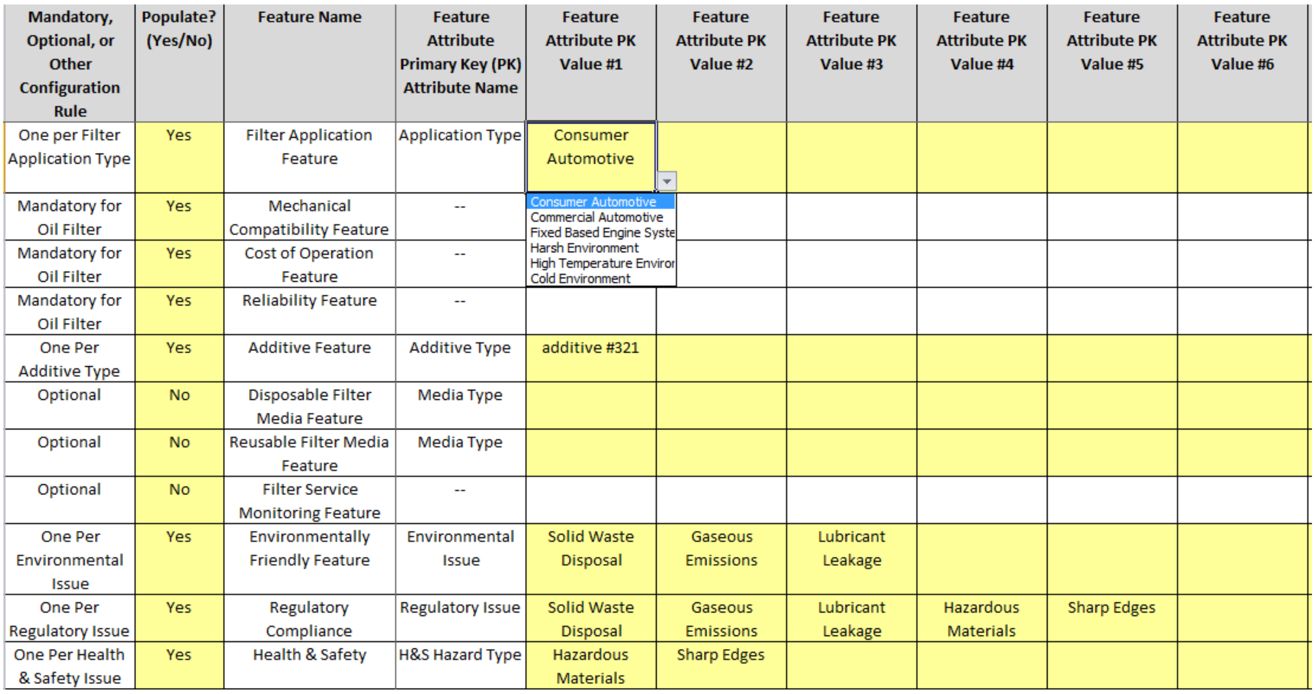
ESD Environment States: created: 8/27/2014 10:23:56 AM modified: 11/7/2014 12:39:31 PM 103% 1059 x 717

Figure 11: State Model, Including Interactions

For a project, each applicable S*Pattern (Enterprise, Product, Manufacturing System, etc.) is configured to specific S*Models applicable in that case. This may be performed on any information system (COTS modelling tool, PLM system, etc.) that has been mapped to the S*Metamodel and set up with an S*Configuration Agent algorithm. For example, the process of configuring Oil Filter Product Pattern Features is shown in Figure 12, and the resulting configured System Requirements are shown in Figure 13.

The Manufacturing System Pattern

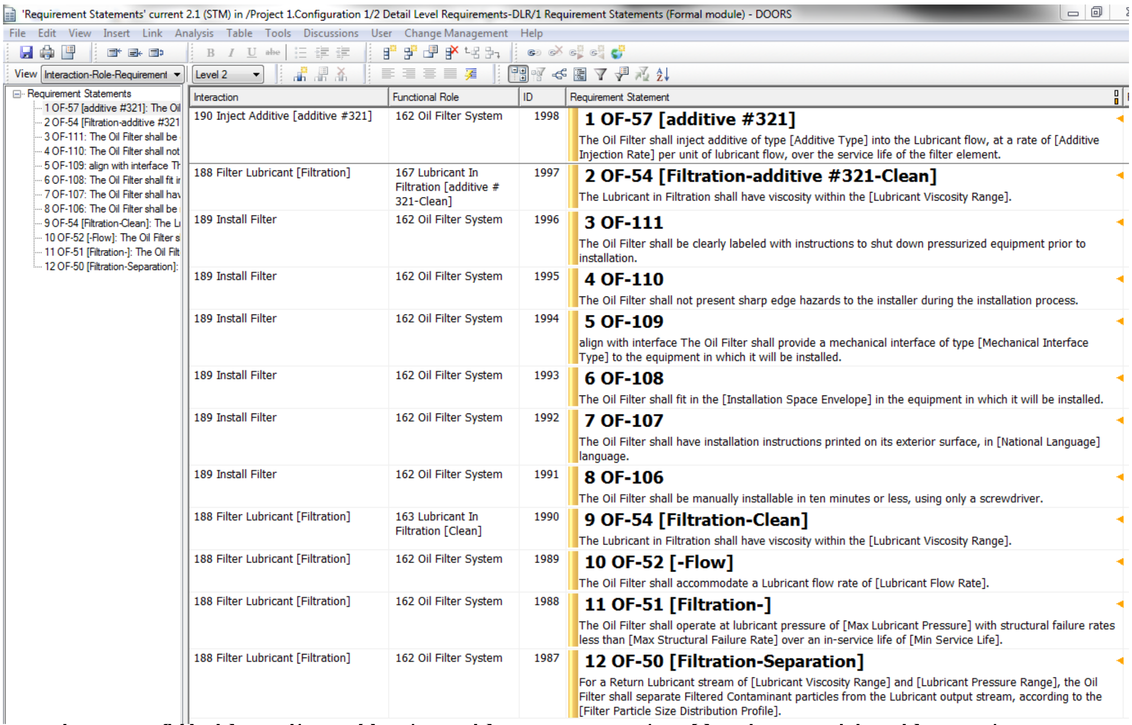
The Manufacturing System Pattern, another S*Pattern, describes the enterprise's manufacturing systems (processes, equipment, controls, people, facilities, materials). This pattern also includes all the S*Metamodel aspects, including those summarized by Figure 1, and having views similar to the preceding Product model series. Like the S*Patterns described above, the Manufacturing System Pattern is managed in some modelling environment, separately configured for each manufacturing system configuration or project, and those configured S*Models are suitable to be managed in a PLM, modelling tool, or similar information system. For some enterprises, it is most efficient for the scope of this pattern to include the product's packaging systems—consumer packaged products and pharmaceuticals are typical examples. Some of the principles of Manufacturing and Packaging System Patterns are described in (Bradley et al 2010, and Schindel 2012b).



Mandatory, Optional, or Other Configuration Rule	Populate? (Yes/No)	Feature Name	Feature Attribute Primary Key (PK) Attribute Name	Feature Attribute PK Value #1	Feature Attribute PK Value #2	Feature Attribute PK Value #3	Feature Attribute PK Value #4	Feature Attribute PK Value #5	Feature Attribute PK Value #6
One per Filter Application Type	Yes	Filter Application Feature	Application Type	Consumer Automotive					
Mandatory for Oil Filter	Yes	Mechanical Compatibility Feature	--						
Mandatory for Oil Filter	Yes	Cost of Operation Feature	--						
Mandatory for Oil Filter	Yes	Reliability Feature	--						
One Per Additive Type	Yes	Additive Feature	Additive Type	additive #321					
Optional	No	Disposable Filter Media Feature	Media Type						
Optional	No	Reusable Filter Media Feature	Media Type						
Optional	No	Filter Service Monitoring Feature	--						
One Per Environmental Issue	Yes	Environmentally Friendly Feature	Environmental Issue	Solid Waste Disposal	Gaseous Emissions	Lubricant Leakage			
One Per Regulatory Issue	Yes	Regulatory Compliance	Regulatory Issue	Solid Waste Disposal	Gaseous Emissions	Lubricant Leakage	Hazardous Materials	Sharp Edges	
One Per Health & Safety Issue	Yes	Health & Safety	H&S Hazard Type	Hazardous Materials	Sharp Edges				

Consumer Automotive
Commercial Automotive
Fixed Based Engine Syste
Harsh Environment
High Temperature Enviror
Cold Environment

Figure 12: Configuring Pattern Features, to Generate Configured System Model



'Requirement Statements' current 2.1 (STM) in /Project 1.Configuration 1/2 Detail Level Requirements-DLR/1 Requirement Statements (Formal module) - DOORS

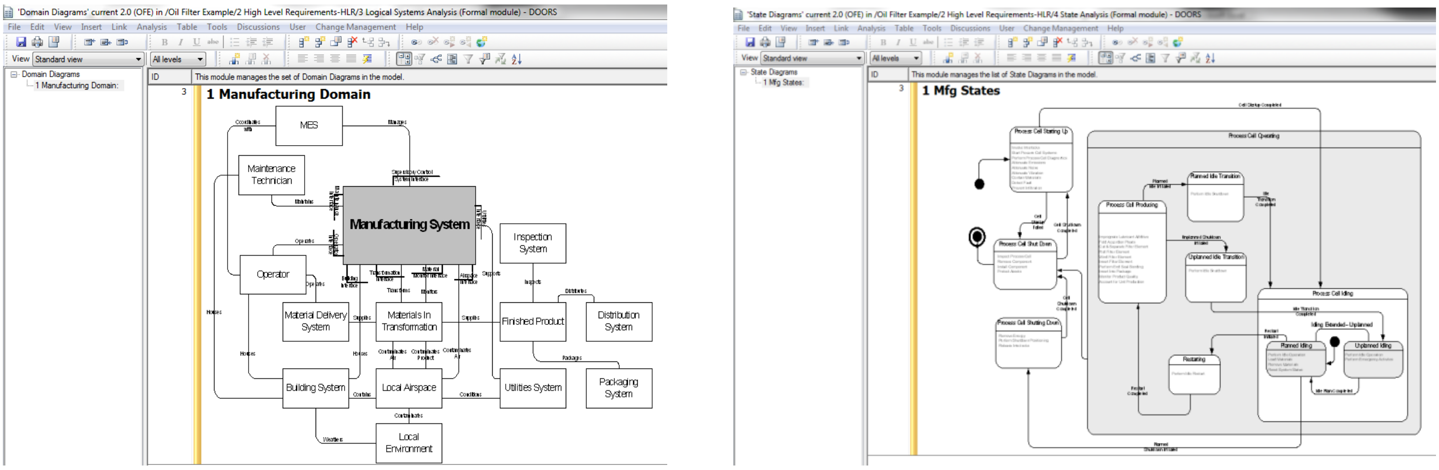
File Edit View Insert Link Analysis Table Tools Discussions User Change Management Help
View Interaction-Role-Requirement
Level 2

Requirement Statements
1 OF-57 [additive #321]: The Oil
2 OF-54 [Filtration-additive #321
3 OF-111: The Oil Filter shall be
4 OF-110: The Oil Filter shall not
5 OF-109: align with interface Th
6 OF-108: The Oil Filter shall fit in
7 OF-107: The Oil Filter shall hav
8 OF-106: The Oil Filter shall be
9 OF-54 [Filtration-Clean]: The Lu
10 OF-52 [-Flow]: The Oil Filter s
11 OF-51 [Filtration-]: The Oil Filt
12 OF-50 [Filtration-Separation]:

Interaction	Functional Role	ID	Requirement Statement
190 Inject Additive [additive #321]	162 Oil Filter System	1998	**1 OF-57 [additive #321]** The Oil Filter shall inject additive of type [Additive Type] into the Lubricant flow, at a rate of [Additive Injection Rate] per unit of lubricant flow, over the service life of the filter element.
188 Filter Lubricant [Filtration]	167 Lubricant In Filtration [additive #321-Clean]	1997	**2 OF-54 [Filtration-additive #321-Clean]** The Lubricant in Filtration shall have viscosity within the [Lubricant Viscosity Range].
189 Install Filter	162 Oil Filter System	1996	**3 OF-111** The Oil Filter shall be clearly labeled with instructions to shut down pressurized equipment prior to installation.
189 Install Filter	162 Oil Filter System	1995	**4 OF-110** The Oil Filter shall not present sharp edge hazards to the installer during the installation process.
189 Install Filter	162 Oil Filter System	1994	**5 OF-109** align with interface The Oil Filter shall provide a mechanical interface of type [Mechanical Interface Type] to the equipment in which it will be installed.
189 Install Filter	162 Oil Filter System	1993	**6 OF-108** The Oil Filter shall fit in the [Installation Space Envelope] in the equipment in which it will be installed.
189 Install Filter	162 Oil Filter System	1992	**7 OF-107** The Oil Filter shall have installation instructions printed on its exterior surface, in [National Language] language.
189 Install Filter	162 Oil Filter System	1991	**8 OF-106** The Oil Filter shall be manually installable in ten minutes or less, using only a screwdriver.
188 Filter Lubricant [Filtration]	163 Lubricant In Filtration [Clean]	1990	**9 OF-54 [Filtration-Clean]** The Lubricant in Filtration shall have viscosity within the [Lubricant Viscosity Range].
188 Filter Lubricant [Filtration]	162 Oil Filter System	1989	**10 OF-52 [-Flow]** The Oil Filter shall accommodate a Lubricant flow rate of [Lubricant Flow Rate].
188 Filter Lubricant [Filtration]	162 Oil Filter System	1988	**11 OF-51 [Filtration-]** The Oil Filter shall operate at lubricant pressure of [Max Lubricant Pressure] with structural failure rates less than [Max Structural Failure Rate] over an in-service life of [Min Service Life].
188 Filter Lubricant [Filtration]	162 Oil Filter System	1987	**12 OF-50 [Filtration-Separation]** For a Return Lubricant stream of [Lubricant Viscosity Range] and [Lubricant Pressure Range], the Oil Filter shall separate Filtered Contaminant particles from the Lubricant output stream, according to the [Filter Particle Size Distribution Profile].

Figure 13: Resulting System Requirements, Configured for Project

A subset of the views typical of S*Models are illustrated in Figure 14 for an example family of manufacturing systems—those which produce the Oil Filter Product Line. The views shown in this section illustrate the use of another COTS engineering tool, again mapped to the S*Metamodel, so it can store the same compatible model data. There are equivalents if other engineering tools are used:



'Domain Diagrams' current 2.0 (OFE) in /Oil Filter Example/2 High Level Requirements-HLR/3 Logical Systems Analysis (Formal module) - DOORS

View: Standard view; All levels

Domain Diagrams
1 Manufacturing Domain:

ID | This module manages the set of Domain Diagrams in the model.

3 **1 Manufacturing Domain**

'State Diagrams' current 2.0 (OFE) in /Oil Filter Example/2 High Level Requirements-HLR/4 State Analysis (Formal module) - DOORS

View: Standard view; All levels

State Diagrams
1 Mfg States:

ID | This module manages the list of State Diagrams in the model.

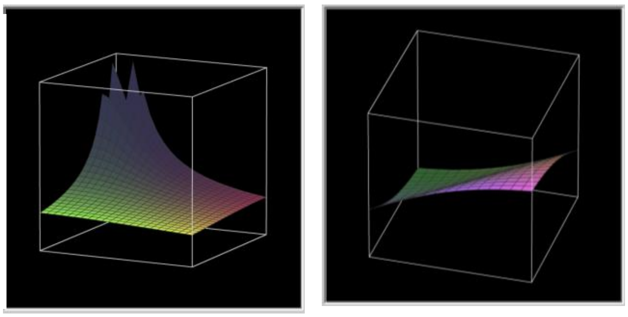
3 **1 Mfg States**

Figure 14: Manufacturing System Domain and State Model Views

The Perform End Seal Bonding interaction of the example Manufacturing System Pattern is based on the transfer function modelling principles of (Schindel 2005b) and the manufacturing transformation principles of (Schindel 2012b). Figure 15 illustrates model views of two attribute couplings associated with this interaction and a later product life cycle interaction:

Unit Throughput as a function of Heat Time and Spray Time

X-Axis (Horizontal 1):
Heat Time
Y-Axis (Horizontal 2):
Spray Time
Z-Axis (Vertical):
Unit Throughput



Additive Life as a function of Heat Time and Spray Time

X-Axis (Horizontal 1):
Heat Time
Y-Axis (Horizontal 2):
Spray Time
Z-Axis (Vertical):
Additive Life

Figure 15: Manufacturing System Interaction Attribute Coupling Map Views

The System of Innovation Pattern

Referring to the R&D enterprise subsystem of Figure 5, the System of Innovation Pattern is another S*Pattern, describing the enterprise's system of innovation for creating new or modified configurations of all the other enterprise subsystems shown in Figure 5. It thus includes product development, but also manufacturing process development and equipment engineering, distribution, and other aspects. This pattern also includes all the S*Metamodel aspects, including those summarized by Figure 1. Like the Enterprise Pattern described above, the SOI Pattern is managed in some modelling environment, separately configured for each innovation project, and those configured S*Models are suitable to be managed in a PLM, modelling tool, or similar information system.

The System of Innovation (R&D) Pattern returns us to Figure 3, which summarizes one "slice" of that pattern—the familiar Systems Engineering "Vee", including appearances of each of the processes of (ISO 15288 2008, 2014). The System of Innovation Pattern is in fact a formal S*Model of ISO15288, so the "Vee" view is only one informal high level summary of a more explicit model. For example, Figure 16 shows progressive details of the SOI Pattern for the Verification Process of ISO 15288.

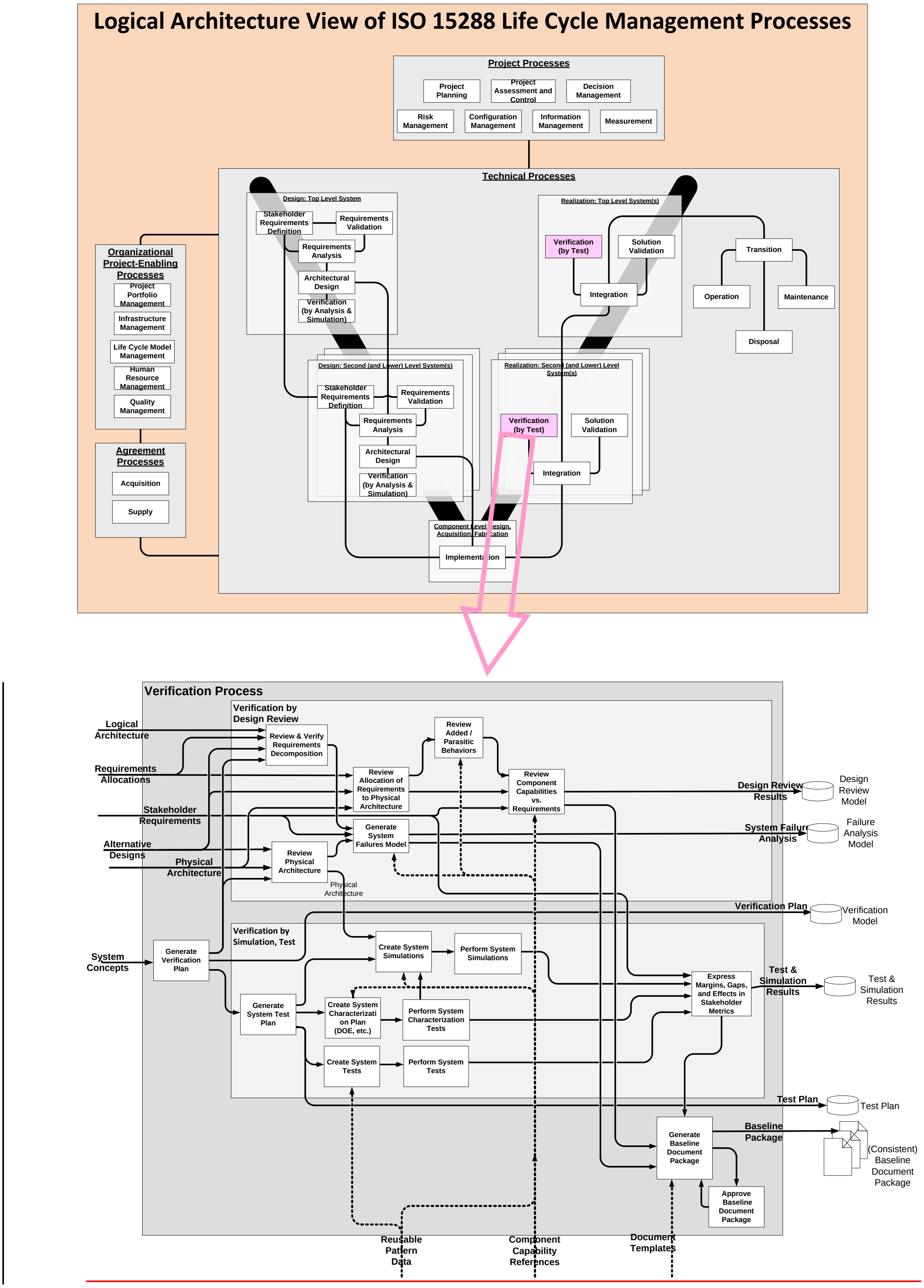

Logical Architecture View of ISO 15288 Life Cycle Management Processes
Project Processes
Project Planning
Project Assessment and Control
Decision Management
Risk Management
Configuration Management
Information Management
Measurement
Technical Processes
Organizational Project-Enabling Processes
Project Portfolio Management
Infrastructure Management
Life Cycle Model Management
Human Resource Management
Quality Management
Agreement Processes
Acquisition
Supply
Design: Top Level System
Stakeholder Requirements Definition
Requirements Validation
Requirements Analysis
Architectural Design
Verification (by Analysis & Simulation)
Realization: Top Level System(s)
Verification (by Test)
Solution Validation
Integration
Transition
Operation
Maintenance
Disposal
Design: Second (and Lower) Level System(s)
Stakeholder Requirements Definition
Requirements Validation
Requirements Analysis
Architectural Design
Verification (by Analysis & Simulation)
Realization: Second (and Lower) Level System(s)
Verification (by Test)
Solution Validation
Integration
Component Level Design, Acquisition, Fabrication
Implementation
Verification Process
Verification by Design Review
Logical Architecture
Requirements Allocations
Stakeholder Requirements
Alternative Designs
Physical Architecture
Review & Verify Requirements Decomposition
Review Allocation of Requirements to Physical Architecture
Review Added / Parasitic Behaviors
Review Component Capabilities vs. Requirements
Generate System Failures Model
Review Physical Architecture
Physical Architecture
Design Review Results
Design Review Model
System Failure Analysis
Failure Analysis Model
Verification Plan
Verification Model
System Concepts
Generate Verification Plan
Verification by Simulation, Test
Create System Simulations
Perform System Simulations
Generate System Test Plan
Create System Characterization Plan (DOE, etc.)
Perform System Characterization Tests
Create System Tests
Perform System Tests
Express Margins, Gaps, and Effects in Stakeholder Metrics
Test & Simulation Results
Test & Simulation Results
Test Plan
Test Plan
Generate Baseline Document Package
Baseline Package
(Consistent) Baseline Document Package
Approve Baseline Document Package
Reusable Pattern Data
Component Capability References
Document Templates

Figure 16: Example Drill-down Into System of Innovation Pattern—The Verification Process Model

A key aspect of the SOI Pattern is that it explicitly recognizes both MBSE and Pattern-Based methods. For example, Figure 16 shows the use of configurable patterns of system verification—represented as configurable pattern data entering from the bottom of Figure 16. This is further discussed in (Cook et al 2015) and (Nolan, et al 2015).

The System of Innovation Pattern includes roles played by human and automated agents, across the life cycle of systems. These include activities associated with diverse existing COTS automation tools, including (SysML or other) modeling tools, requirements management databases, and PLM systems from multiple suppliers. As shown in Figure 17, an S*Metamodel schema map (profile) is provided for each such system, so that they can uniformly represent project-specific configured S*Models and generalized S*Patterns. S*Configuration Process agents likewise provide a unified approach to configuring S*Models from S*Patterns.

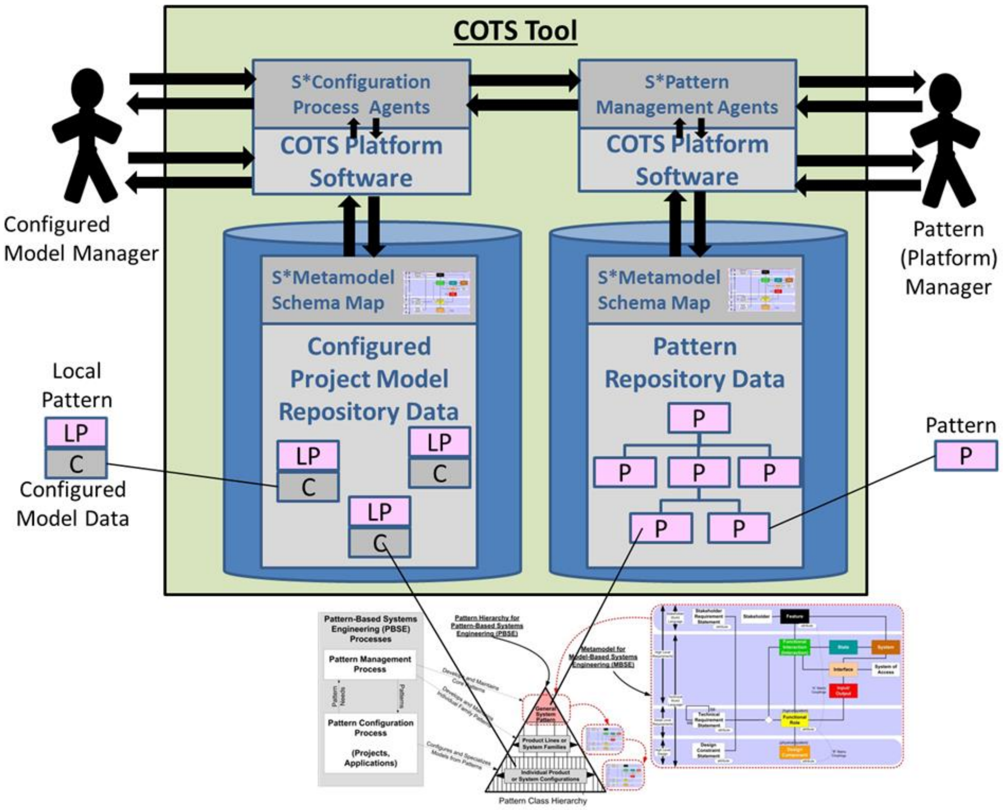

COTS Tool
S*Configuration Process Agents
COTS Platform Software
S*Pattern Management Agents
COTS Platform Software
Configured Model Manager
Pattern (Platform) Manager
S*Metamodel Schema Map
Configured Project Model Repository Data
S*Metamodel Schema Map
Pattern Repository Data
Local Pattern
LP
C
Configured Model Data
Pattern
P
Pattern-Based Systems Engineering (PBSE) Processes
Pattern Management Process
Pattern Needs
Patterns
Pattern Configuration Process
(Projects, Applications)
Pattern Hierarchy for Pattern-Based Systems Engineering (PBSE)
Develop and Maintain Core Patterns
Develop and Maintain Individual Family Patterns
Configures and Specializes Models from Patterns
General System Pattern
Product Lines or System Families
Individual Product or System Configurations
Pattern Class Hierarchy
Metamodel for Model-Based Systems Engineering (MBSE)
Stakeholder Requirement Statement
Stakeholder
Feature
Functional Interaction (Interaction)
State
System
Interface
System of Access
Input/ Output
Technical Requirement Statement
Functional Role
Design Constraint Statement
Design Component

Figure 17: Existing COTS Engineering & Modeling Tools, PLM Systems Can All Support Common Underlying S*Metamodel, Innovation Processes

Many third-party COTS tools and information systems provide some means of data exchange among them, using standards-based or other types of exchange interfaces. The approach described here goes further, by providing a deeper underlying semantic compatibility between these existing systems, while still taking advantage of the available exchange interfaces. This is more than an information technology approach, as further aligns the semantics of how human users of these systems conceive of the information they manage. As illustrated in this paper, such approaches have been taken to further leveraging the power of existing COTS systems such as Siemens Teamcenter®, Dassault Systemes ENOVIA®, IBM Rational DOORS®, and SysML® tools such as Sparx Systems Enterprise Architect® and IBM Rational Rhapsody® Architect. Along with their human users, these play Management System (MTS) roles in the hierarchy of Figure 6, integrated within the Enterprise Pattern of Figure 5, for (ISO 15288) specialized work processes, views, and artifacts described by a configured System of Innovation Pattern, such as those in Figure 16.

Example: Integrating Product Development and Production

The explicit physical interactions structure of the S*Metamodel guarantees that each case of enterprise "silo" problems will be visible in the model, associated with boundary-crossing interactions and the emergent behavior that the resulting interaction demonstrates. An example is Product Application Domain interactions for an in-service Oil Filter System product (e.g., Filter Lubricant, Inject Additive) and Production Domain interactions (e.g., Perform End Seal Bonding, Impregnate Lubricant Additives). The attribute couplings of Figure 15 capture the impact of production rates, pressures, temperatures, and raw material characteristics on in-service product reliability, pressure rating, and life. An integrated framework for negotiating and optimizing these across Process Engineering and Product Design is the result. Within that framework we have demonstrated generation of high quality configured product requirements an order of magnitude faster than traditional methods.

Summary and Conclusions

MBSE in general, and model-based patterns (PBSE) in particular, not only apply across the enterprise—they can directly address enterprise-level challenges that arise out of interactions of lower-level enterprise subsystems. The expressive power of explicit models is further leveraged when they do not have to be developed "from scratch" for each project, but can be derived from patterns that themselves accumulate learning as it occurs, becoming a new form of IP, increasing the agility of the enterprise. This changes the perspective of individuals from "learn modeling" to "learn the model" (referring to the enterprise's MBSE pattern IP)—a different perspective from the more popular "learn how to model" movement. In addition to improving the power and capabilities of individuals, existing and in-service engineering modeling and simulation tools, databases, and PLM systems likewise have their power increased when they are enabled to accommodate the stronger semantics of the S*Metamodel.

References

1. (Alexander 1977) Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S., *A Pattern Language*. Oxford University Press, New York, 1977.
2. (Berg 2014) Berg, E., "Affordable Systems Engineering: An Application of Model-Based System Patterns To Consumer Packaged Goods Products, Manufacturing, and Distribution", at INCOSE IW2014 MBSE Workshop, 2014.
3. (Bradley, Hughes, Schindel 2010) Bradley, J., Hughes, M. and Schindel, W., "Optimizing Delivery of Global Pharmaceutical Packaging Solutions, Using Systems Engineering Patterns" Proceedings of the INCOSE 2010 International Symposium (2010).
4. (Cloutier 2008) Cloutier, R., Applicability of Patterns to Architecting Complex Systems: Making Implicit Knowledge Explicit. VDM Verlag Dr. Müller. 2008.
5. (Cook, Schindel 2015) Cook, D., and Schindel, W., "Utilizing MBSE Patterns to Accelerate System Verification", to appear in *Proc. of the INCOSE 2015 International Symposium*, Seattle, WA, July, 2015.
6. (Dove, LaBarge 2014) Dove, R., LaBarge, R., "Fundamentals of Agile Systems Engineering—Part 1" and "Part 2", INCOSE IS2014, July, 2014.
7. (Dove, Schindel 2015) Dove, R., and Schindel, W., "Agile Modeling and Modeling Agile Systems", to appear at INCOSE IW2015 MBSE Workshop, Torrance, CA, January 24, 2015.
8. (Estafan 2008) Estafan, J. 2008. Survey of model-based systems engineering (MBSE) methodologies. INCOSE MBSE Initiative.
9. (Gamma et al 1995) Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Publishing Company, Reading, MA, 1995.

10. (INCOSE Handbook 2014) INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Version 4, International Council on Systems Engineering (2014).
11. (INCOSE Patterns Team 2014) INCOSE/OMG MBSE Initiative: Patterns Challenge Team 2013-14 Web Site: http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
12. (ISO 15288 2014) ISO/IEC 15288: Systems Engineering—System Life Cycle Processes. International Standards Organization (2014).
13. (ISO 26550 2013) ISO/IEC 26550 "Software and Systems Engineering—Reference Model for Product Line Engineering and Management", 2013.
14. (ISO 42010 2011) ISO/IEC/IEEE 42010 "Systems and Software Engineering—Architecture Description", 2011.
15. (Nolan, Pickard, Russell, Schindel 2015) Nolan, A., Pickard, A., Russell, J., Schindel, W., "When two is good company, but more is not a crowd", to appear in *Proc. of the INCOSE 2015 International Symposium*, Seattle, WA, July, 2015.
16. (Peterson, Schindel 2015) Peterson, T., Schindel, W., "Unmanned Ground Vehicle Platforms and Model-Based System Patterns: An Example", to appear in *Proc. of the INCOSE 2015 International Symposium*, Seattle, WA, July, 2015.
17. (Schindel 2005a) Schindel, W., "Pattern-Based Systems Engineering: An Extension of Model-Based SE", INCOSE IS2005 Tutorial TIES 4, (2005).
18. (Schindel 2005b) Schindel, W. "Requirements statements are transfer functions: An insight from model-based systems engineering", Proceedings of INCOSE 2005 International Symposium, (2005).
19. (Schindel 2010) Schindel, W., "Failure Analysis: Insights from Model-Based Systems Engineering", INCOSE International Symposium, Chicago, 2010.
20. (Schindel 2011a) Schindel, W. "Innovation as Emergence: Hybrid Agent Enablers for Evolutionary Competence" in *Complex Adaptive Systems, Volume 1*, Cihan H. Dagli, Editor in Chief, Elsevier, 2011
21. (Schindel 2011b) Schindel, W. "What Is the Smallest Model of a System?", Proc. of the INCOSE 2011 International Symposium, International Council on Systems Engineering (2011).
22. (Schindel 2011c) Schindel, W., "The Impact of 'Dark Patterns' On Uncertainty: Enhancing Adaptability In The Systems World", in Proc. of INCOSE Great Lakes 2011 Regional Conference on Systems Engineering, Dearborn, MI, 2011
23. (Schindel 2012a) Schindel, W. "Introduction to Pattern-Based Systems Engineering (PBSE)", INCOSE Finger Lakes Chapter Webinar, April 26, 2012.
24. (Schindel 2012 b) Schindel, W., "Integrating Materials, Process, & Product Portfolios: Lessons from Pattern-Based Systems Engineering", in *Proc. of Society for Advancement of Materials and Process Engineering* (SAMPE), 2012
25. (Schindel 2013a) Schindel, W. "Interactions: At the Heart of Systems", INCOSE Great Lakes Regional Conference on Systems Engineering, W. Lafayette, IN, October, 2013.
26. (Schindel 2013b) Schindel, W., "Systems of Innovation II: The Emergence of Purpose", *Proceedings of INCOSE 2013 International Symposium* (2013).
27. (Schindel 2014) Schindel, W. "The Difference Between Whole-System Patterns and Component Patterns: Managing Platforms and Domain Systems Using PBSE", INCOSE Great Lakes Regional Conference on Systems Engineering, Schaumburg, IL, October, 2014
28. (Schindel 2015a) Schindel, W., "Maps or Itineraries? A Systems Engineering Insight from Ancient Navigators", to appear in *Proc. of the INCOSE 2015 International Symposium*, Seattle, WA, July, 2015.
29. (Schindel 2015b) Schindel, W., "System Life Cycle Trajectories: Tracking Innovation Paths Using System DNA", to appear in *Proc. of the INCOSE 2015 International Symposium*, Seattle, WA, July, 2015.
30. (Schindel, Beihoff 2012) Schindel W., and Beihoff, B., "Systems of Innovation I: Models of Their Health and Pathologies", Proc. of INCOSE International Symposium, 2012.

31. (Schindel, Peffers, Hanson, Ahmed, Kline 2011) Schindel, W., Peffers, S., Hanson, J., Ahmed, J., Kline, W., "All Innovation is Innovation of Systems : An Integrated 3-D Model of Innovation Competencies ", *Proc. of ASEE 2011 Conference,* American Association for Engineering Education, (2011).

32. (Schindel, Peterson 2013) Schindel, W., and Peterson, T. "Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques", in Proc. of INCOSE 2013 International Symposium, Tutorial, June, 2013.

33. (Schindel, Smith 2002) Schindel, W., and Smith, V., "Results of applying a families-of-systems approach to systems engineering of product line families", SAE International, Technical Report 2002-01-3086 (2002).

Biography

William D. (Bill) Schindel is president of ICTT System Sciences. His engineering career began in mil/aero systems with IBM Federal Systems, included faculty service at Rose-Hulman Institute of Technology, and founding of three systems enterprises. Bill co-led a 2013 project on the science of Systems of Innovation in the INCOSE System Science Working Group. He co-leads the Patterns Challenge Team of the OMG/INCOSE MBSE Initiative. Schindel earned B.S. and M.S. degrees in mathematics.

Stephen A. Lewis is a Senior Systems Engineer at ICTT System Sciences in Terre Haute, Indiana, where has worked since 2008. He has served on the planning committees of the 2013 and 2014 INCOSE Great Lakes Regional Conferences. He currently participates in the Patterns Challenge Team of the OMG/INCOSE MBSE Initiative and the INCOSE Regional Healthcare Working Group. Lewis earned the B.S. in Applied Biology, M.S. in Engineering Management, and J.D. in Law.

Jason J. Sherey is a Principal Systems Engineer for ICTT System Sciences. During his 15 years there he has practiced, documented, taught, helped develop, and mentored in the Systematica™ Methodology. He has modeled patterns for a variety of systems, including engines, tractors, trucks, software, business processes, manufacturing systems, medical devices, and guidance systems. He is a past-president of the INCOSE Crossroads of America Chapter, and earned the B.S. in Electrical Engineering, M.S. in Systems Engineering, and M.S. in Engineering Management.

Saumya K. Sanyal leads K2 Firm's Product Lifecycle Management (PLM) services practice. He has over 25 years of experience in EIS, ERP, and PLM processes in business and defense, and has developed acquisition strategies, operational requirements, architectures, and systems. He has identified barriers to change, developed and executed change management action plans, delivered enterprise and business strategies, roadmaps, solution architectures, systems engineering methodologies, processes, and developed multi-corporate project teams. Saumya has graduate degrees in Electrical and Software Engineering.