# Document Tracking Sheet

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Team</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>May 2, 2008</td>
<td>EDM (EDM) Team</td>
<td>Initial Version</td>
</tr>
<tr>
<td>1.1</td>
<td>December 8, 2008</td>
<td>EDM Team</td>
<td>Minor edits made throughout the document</td>
</tr>
<tr>
<td>1.2</td>
<td>January 16, 2009</td>
<td>EDM Team</td>
<td>PD-specific annotations</td>
</tr>
<tr>
<td>1.3</td>
<td>Feb / Mar 2009</td>
<td>EDM Team</td>
<td>Many corrections/ edits/ incorporated review suggestions</td>
</tr>
<tr>
<td>1.4</td>
<td>March 30, 2009</td>
<td>EDM Team</td>
<td>Restructure</td>
</tr>
<tr>
<td>2.0</td>
<td>June 30, 2009</td>
<td>EDM Team</td>
<td>Remove identifiers, clarify identity &amp; independence, add whole/part, add events</td>
</tr>
<tr>
<td>2.1</td>
<td>October 14, 2009</td>
<td>EDM Team</td>
<td>Update Appendices, Update guidance to reflect new concept worksheets allowed under PD 15.</td>
</tr>
<tr>
<td>2.2</td>
<td>October 5, 2010</td>
<td>EDM Team</td>
<td>Update Appendices, Update guidance to reflect modifications to methodology since October 2009.</td>
</tr>
<tr>
<td>2.3</td>
<td>March 15, 2011</td>
<td>EDM Team</td>
<td>Added “Is derivable” property to attribute; 333 documented proper name (is ,adjective&gt;) for Boolean attributes; fixed documentation for common attribute definitions; modified all affected metamodel</td>
</tr>
</tbody>
</table>


extracted figures.

Added Preface
(recognizing Guizzardi’s original inspiration, as well as noting differences between his work and ours)
Table of Contents

Preface .................................................................................................................................................. xii

1 Introduction ........................................................................................................................................ 1
  1.1 Purpose (of this document) ........................................................................................................... 1
  1.2 Intended Uses (of an ELDM) ....................................................................................................... 1
  1.3 Intended Audience (for this document) ....................................................................................... 2
  1.4 Data Model Layers (relevant to these guidelines) ....................................................................... 2
  1.5 Notations (used in an ELDM) .................................................................................................... 3
  1.6 Automated Tool Support ............................................................................................................. 4
    1.6.1 Tool-Independent .................................................................................................................. 4
    1.6.2 Style Choices ........................................................................................................................ 4
  1.7 Sources .......................................................................................................................................... 4
  1.8 Structure (of this document) ........................................................................................................ 5

2 Model Concept ..................................................................................................................................... 9
  2.1 Model Properties ......................................................................................................................... 9
    2.1.1 Common Model Properties .................................................................................................. 10
    2.1.2 Additional Model Properties .............................................................................................. 10
    2.1.3 Model Properties Based on Relationships to other Concepts ......................................... 11

3 Business Area Concept .................................................................................................................... 12
  3.1 Business Area Properties ............................................................................................................ 13
    3.1.1 Common Business Area Properties ................................................................................... 14
    3.1.2 Business Area Properties Based on Relationships to other Concepts ............................... 14

4 Diagram Concept .................................................................................................................................. 17
  4.1 Diagram Properties ...................................................................................................................... 17
### 4.1 Common Diagram Properties
- p. 18

### 4.1.2 Additional Diagram Property
- p. 19

### 4.1.3 Diagram Properties Based on Relationships to other Concepts
- p. 20

### 4.2 Types of Diagrams
- p. 21

### 4.3 Other Diagram Guidelines
- p. 24

### 5 Class (or Entity) Concept
- p. 27

#### 5.1 Class Properties
- p. 27

##### 5.1.1 Common Class Properties
- p. 28

##### 5.1.2 Additional Class Properties
- p. 29

###### 5.1.2.1 Class Principles
- p. 30

####### 5.1.2.1.1 Continuant and Occurrent Classes
- p. 31

####### 5.1.2.1.2 Independent and Dependent Classes
- p. 31

####### 5.1.2.1.3 Principle of Identity of a Class
- p. 32

####### 5.1.2.1.4 Concrete and Abstract Classes
- p. 32

####### 5.1.2.1.5 Realization Classes
- p. 33

####### 5.1.2.1.6 Specialization Classes
- p. 34

####### 5.1.2.1.7 Permanent and Transient Classes
- p. 34

###### 5.1.2.2 Class Stereotype Details
- p. 34

####### 5.1.2.2.1 Kind
- p. 34

####### 5.1.2.2.2 Role
- p. 35

####### 5.1.2.2.3 Category
- p. 37

####### 5.1.2.2.4 Role Category
- p. 38

####### 5.1.2.2.5 Dependent
- p. 40

####### 5.1.2.2.6 Associative
- p. 41

####### 5.1.2.2.7 Event
- p. 43
5.1.2.3 Class Stereotype Guidelines ................................................................. 45
5.1.3 Class Properties Based on Relationships to other Concepts .................. 47

6 Attribute Concept ....................................................................................... 50

6.1 Attribute Properties .................................................................................. 54
6.1.1 Common Attribute Properties .............................................................. 55
6.1.2 Additional Attribute Properties ........................................................... 56
6.1.3 Attribute Properties Based on Relationships to other Concepts .............. 57

7 Relationship Concept .................................................................................. 61

7.1 Relationship Properties ............................................................................ 61
7.1.1 Common Relationship Properties ......................................................... 63
7.1.2 Additional Relationship Properties ...................................................... 63
7.1.2.1 Relationship Details ........................................................................ 63
    7.1.2.1.1 Inheritance Relationship Type ................................................... 64
    7.1.2.1.1.1 Realization Relationship ......................................................... 64
    7.1.2.1.1.2 Specialization Relationship .................................................... 66
    7.1.2.1.1.3 Inheritance Sets and Independent Inheritance ......................... 68
        7.1.2.1.1.3.1 Inheritance Sets .............................................................. 68
        7.1.2.1.1.3.2 Independent Inheritance ................................................. 72
    7.1.2.1.1.4 Inheritance Relationship Properties ....................................... 73
    7.1.2.1.1.5 Special Guidelines and Rules for Inheritance Relationships ....... 74
    7.1.2.1.2 Association Relationship Type .................................................. 75
        7.1.2.1.2.1 Dependency Association ................................................... 75
        7.1.2.1.2.2 Non-dependency Association ............................................. 77
        7.1.2.1.2.3 Whole/Part Associations .................................................... 78
            7.1.2.1.2.3.1 Aggregation Association ............................................. 81
8.2.4.1  Data Structure Data Type Properties: ................................................................. 115
8.2.4.1.1  Common Data Structure Properties................................................................. 115
8.2.4.1.2  Data Structure Data Type Properties Based on Relationships to other Concepts 116
8.2.5  Union Data Type..................................................................................................... 117
8.2.5.1  Union Data Type Properties: .............................................................................. 117
8.2.5.1.1  Common Union Data Type Properties.............................................................. 118
8.2.5.1.2  Union Data Type Properties Based on Relationships to other Concepts .......... 118

APPENDIX A.  ELD M TOOL-INDEPENDENT METAMODEL .............................................. A-1
APPENDIX B.  ELD M COMMON PROPERTIES ................................................................. B-1
APPENDIX C. ELD M NAMING GUIDELINES ................................................................. C-1
APPENDIX D.  ELD M ACRONYM GUIDELINES ............................................................... D-1
APPENDIX E.  ELD M DEFINITION GUIDELINES ............................................................. E-1
APPENDIX F.  ELD M DEFINITION SOURCING GUIDELINES ........................................... F-1
APPENDIX G.  ADDITIONAL ELD M GUIDELINES .......................................................... G-1
APPENDIX H.  OVERVIEW OF ELD M STEREOTYPES ................................................... H-1
APPENDIX I.  Modifications to Guizzardi............................................................I-1
List of Figures and Tables

Figure 1-1 The Relationship between the Two Layers of Data Models .......................................................... 3
Figure 2-1 The Model Concept and Its Properties and Relationships ................................................................. 10
Figure 3-1 The Business Area Concept and Its Properties and Relationships..................................................... 14
Figure 4-1 Diagram Concept and Its Relationships .......................................................................................... 18
Figure 4-2 Example of Overall Diagram Rules .................................................................................................. 23
Table 4-1 Placement, Color and Font for Items on Diagrams ........................................................................... 25
Table 4-2 RGB Settings for Colors .................................................................................................................. 26
Figure 5-1 Class Concept and Its Properties and Relationships .......................................................................... 28
Table 5-1 Class Defining Principles .................................................................................................................. 30
Figure 5-2 Example of a Kind Class - Person ..................................................................................................... 35
Figure 5-3 Example of a Role Class - Doctor ..................................................................................................... 36
Figure 5-4 Example of a Category Class – Party ................................................................................................ 38
Figure 5-5 Example of a Role Category Class – Customer ................................................................................. 40
Figure 5-6 Example of a Dependent Class – Person Name ................................................................................ 41
Figure 5-7 Example of an Associative Class – Report Person Usage ................................................................. 42
Figure 5-8 An Example of an Associative Class That Does Not Resolve a Many-To-Many Relationship ... 43
Figure 5-9: Employment Event Example .......................................................................................................... 45
Figure 5-10 Class Type Decision Tree .............................................................................................................. 45
Figure 5-11 Set of Questions to Determine the Correct Stereotype ................................................................. 46
Figure 6-1 Modeling an Attribute as a Separate and Related Class Initially .................................................. 50
Figure 6-2 Bringing the Attribute into the Class ............................................................................................... 51
Figure 6-3 Modeling Manufacturer as a Role of Organization ....................................................................... 52
Table 6-1: Tests to Determine Whether Candidate Attribute Should be an Attribute or a Separate Class 52
Data Modeling Guide (DMG) For An Enterprise Logical Data Model, V2.3; 15 March 2011

Figure A-1 Tool-Independent Metamodel for an ELDM ............................................................. A-2
Figure B-1 Common Properties That Apply to ELDM Concepts.................................................. B-1
Table B-1 Common Properties Applied to ELDM Concepts......................................................... B-2
Table C-1 Examples Showing Preferred Way to Name Attributes................................................ C-5
Table C-2 Common Attribute Names and Suggested Definitions/Sources...................................... C-5
Table E-1 Characteristics of a “Wooden Pencil” .......................................................................... E-1
Figure E- 1: Defining a <<kind>> Class ................................................................................... E-6
Figure E- 2: Defining a <<role>> Class .................................................................................... E-7
Figure E- 3: Defining a <<category>> Class ............................................................................. E-7
Figure E-4: Defining a <<role category>> Class ........................................................................ E-8
Figure E-5: Defining a <<role category>> Class, Option 2 ....................................................... E-9
Figure E-6: Defining a <<dependent>> Class ............................................................................ E-10
Figure E-7: Defining an <<associative>> Class ......................................................................... E-10
Figure E-8: Defining an <<event>> Class .................................................................................. E-11
Figure E-9: Defining an <<enum>> Class ................................................................................... E-12
Figure E-10: Defining a <<struct>> Data Type ......................................................................... E-13
Figure E-11: Defining a <<union>> Data Type ......................................................................... E-13
Figure F- 1: Decision Tree to Select “Primary” Source ........................................................... F-5
Figure G- 1: Apparent XOR Boolean Constraint Scenario ..................................................... G-3
Figure G- 2: Approach to Avoid Using Boolean Constraints .................................................. G-4
Figure G- 3: Sample Metadata Class ..................................................................................... G-5
Figure G- 4: The Metadata Business Rule Concept and Its Properties and Relationships ........ G-6
Table H-1 Stereotypes for an ELDM ...................................................................................... H-1
Figure I-1: Original Guizzardi Constructs intentionally omitted from methodology ................ I-1
Preface

The success of this Data Modeling Guide for an Enterprise Logical Data Model could not have been possible without the inspired and exhaustive research of Giancarlo Guizzardi, notably his “Ontological Foundations for Structural Conceptual Models,” published in 2005 in association with the Centre for Telemetrics and Information Technology, which provided the theoretical foundation for the methodologies describe within, and from which real world, practical implementations have already ensued.

At the core of Guizzardi’s modeling paradigm are the principles of Rigidity, Uniform Identity and Existential Dependence. From those foundational tenets he extrapolates the concepts of SortalUniversal (Unified Principle of Identity), MixinUniversal (Disparate Set of Concepts), and finally the constructs of SubstanceSortal (Kind, Quantity, and Collective), Subkind, Phase, Role, Category, RoleMixin and Mixin. In short, the total package offered to us by Guizzardi contained a complete and fully integrated set of concepts and constructs that left us wanting for nothing.

It is fascinating now to reflect on the journey we took to bring the fullness of Guizzardi’s unique insights to fruition. Indeed while our initial intent was to embrace the entirety of his vision and implement it intact, we found ourselves, not unexpectedly, confronted with some real world exigencies that gave us pause to consider and ultimately incorporate, if ever so slightly, some modeling compromises as well as include some notable augmentations.

The impetus for the small compromises, indeed more like shortcuts, was to neutralize some criticisms from our modelers that the methodology was perhaps too rich and too full of choices that, while perspicacious indeed, were ultimately not needed to represent the range and vagaries of our particular domain. The impetus for our augmentations was to extend the principles to a wide range of modeling needs.

Our customization endeavor announced itself smoothly with two distinct efforts. Our first effort was to pare and focus the modeling terminology itself, that is, to rename, combine or defer a few terms. For example, we chose to rename RoleMixin to RoleCategory, Relator to Associative, as well as altogether to not use SubKind, Phase, Quantity and Collective. Our second effort was to incorporate some additional, somewhat more traditional, design techniques, notably, Business Rules, Partitioning Attributes and an exhaustive set of Data Types in order to automate the derivation of subsequent models, namely our Domain Schemas as well as Physical Database Models. In this way, we consider our efforts to be a valid augmentation of the original work.

Finally, we took liberties with the use of the terms “conceptual” versus “logical.” Whereas Guizzardi uses “conceptual” when referring to the models produced by his methods, we chose instead to keep “logical” in our title for a practical reason. Our enterprise model contains a tremendous amount of detail compared to the traditional conceptual models (most familiar to our database savvy audience) in which much of the detail is suppressed to present a high level view of the major constructs to a non-technical audience. We chose not to use the term “conceptual” in the fear it would be misleading to our potential readership.
At the end of the day, we believe we not only retained the full benefit of the immanent breadth and cohesion of Guizzardi’s original work but indeed augmented it along the way and thus at once assuaged the theoretical and practical sensibilities of our entire team at all levels of modeling abstraction.
1 Introduction

1.1 Purpose (of this document)

The primary purpose of this document is to provide guidance for developing an Enterprise Logical Data Model (ELDM).

1.2 Intended Uses (of an ELDM)

While an ELDM may serve many purposes in an enterprise, the primary driver for the development of this methodology was to enable data interoperability and effective data sharing. An ELDM, being a single logical model of an enterprise’s data, defines and labels data concepts and elements, describes the relationships between the data concepts, and expresses the business rules that apply to that data. This methodology describes how to express a single ELDM from which many physical implementations can be derived. These physical implementations carry forward the common definitions and labels, relationships, and business rules while incorporating local design optimizations. This methodology was developed with a focus on the following physical implementations involved in data interoperability and sharing:

- Physical Data Models specifying the structures and formats required for optimal design of physical schemas including:
  - Data exchange formats expressed in a variety of languages such as eXtensible Markup Language (XML) Schema Definitions (XSDs)
  - Resource Description Framework Schemas (RDFS)
  - Database schemas (for any relational database).

However, the authors foresee a wider application of an ELDM to an enterprise’s data needs including:

- Providing a basis for:
  - Query brokers and federated queries (i.e., that search through multiple data repositories looking for common data concepts), by representing the “common” view of data that exists in multiple physical repositories
  - Data structures and data objects encoded in software applications
  - System and service application programming interfaces (APIs) and interface control documents (ICDs)
  - Information exchanges between service-oriented architecture components;
  - Mappings of legacy databases (to the data concepts in an ELDM)
  - Providing a basis for assessing architectural compliance of solutions (that use data).
• Adding semantics to data by defining data elements and their relationships (for any purpose) in a user-friendly, understandable and standard way
• Facilitating communication among user communities by providing a common vocabulary of data
• Specifying data-related requirements for new solutions
• Accelerating the development of and reducing life-cycle costs of automated solutions by reusing the data structures and definitions made available in an ELDM.

1.3 Intended Audience (for this document)
This document is intended for anyone who wants to add, change, or review any portion of an ELDM that is developed according to these guidelines. In particular, this document is intended for enterprise data modelers and their Subject Matter Experts (SMEs) to help them accurately depict and define information used by an enterprise.

In addition, this document is aimed at:

• Those who are concerned about and desire to improve data management within an enterprise
• The numerous other stakeholders concerned about the development and management of automated solutions within an enterprise.

1.4 Data Model Layers (relevant to these guidelines)
There are two layers of data models that need to be understood in order to take full advantage of an enterprise data model developed according to these guidelines.

1. An ELDM – The top layer is a single ELDM, which depicts all of the conceptual elements related to the data needed and used by the enterprise to conduct its business. The elements of this model are defined in business terms, taking all perspectives of the data in the enterprise into account. This top layer is the subject of this data modeling guide.

2. PDMs for Data Exchange or Storage Based on an ELDM – The lower layer consists of multiple PDMs, each derived from or based on the single ELDM and used to engineer automated solutions of at least two different types: those designed to facilitate the exchange of data (using XML documents that conform to XML schema descriptions or other exchange languages) and those designed to facilitate the storage of data (in relational databases using DDL). Regardless of the type of automated solution, each physical data model in this layer:
   o Consists of a subset of the data in the ELDM.
   o Maintains a link to the business specification in the ELDM.
   o May structure the data differently from the ELDM in order to optimize its solution.
Data Modeling Guide (DMG) For An Enterprise Logical Data Model, V2.3; 15 March 2011

- Adds more detailed specifications about the data that are required for a particular implementation. Examples of additional details that may get specified in a PDM include: primary (key) identifiers, field lengths, whether an attribute should be an element or attribute in the XSD, whether a class is abstract, XSD encoding choices for classes, attributes or relationships; which classes act as a “root” class for the XSD; indexes, etc.

Figure 1-1 depicts the relationship between the two layers of data models.

![Diagram showing the relationship between Enterprise LDM and PDM for Data Exchange and PDM for Data Storage.]

Figure 1-1 The Relationship between the Two Layers of Data Models

Having automated solutions throughout the enterprise for both the exchange and storage of data that are based on the same semantics greatly facilitates data interoperability and the integration among automated solutions.

1.5 Notations (used in an ELDM)

Many data modelers have a preference for a particular data modeling notation. Many prefer an Entity Relationship Diagram (ERD) notation and still others prefer an Object-Oriented Unified Modeling Language (UML) notation. The methodology described by this Data Modeling Guide (DMG) can easily be used by data modelers using their preferred notation, ERD or UML. For ease of reading, this DMG depicts all examples using UML notation.

Both of these standard notations needed to be extended (with additional properties) to some degree in order to support all of the concepts included in this methodology. Some of these extensions will be made visible on diagrams and others will not. In general, extensions to the ERD and UML notations were included (or properties were added) for one of the following two reasons:
1. To provide a property in an ELDM to control a characteristic of a physical data structure based on the ELDM (Note: this was the dominant consideration) or

2. To provide semantic clarity of the data (e.g., by adding a stereotype of a class to indicate that it represents a “role”).

1.6 Automated Tool Support

1.6.1 Tool-Independent

Most of the guidelines, conventions and notations described in this document are tool independent, meaning that any data modeling tool would need to have some way to support them. For consistency purposes, however, all of the data model examples (i.e., figures) provided in this document have been produced using the Sybase PowerDesigner (PD) data modeling tool.

Note: Hidden text has been embedded into this document to provide additional guidance for users of Power Designer and for certain local users. If the next text after this paragraph is “1.6.2 Style Choices”, then that additional guidance is not being displayed or printed. Please refer to Microsoft Word documentation if you desire to display or print the additional guidance. If the next text is “PD Guidance 1” then the additional guidance is being displayed. Please refer to Microsoft Word documentation if you desire to hide or not print the additional guidance.

PD Guidance 1

The original version of this document was written to support this tool-independent view of the new ELDM methodology. To realize this methodology, a specific set of implementation standards has been developed using Sybase PD extensions to produce UML models. The version of the ELDM DMG you are reading uses text boxes such as this one, numbered “PD Guidance n”, to provide additional guidance for applying this tool-specific implementation.

Local Guidance 1

This version of the ELDM DMG also provides local guidance with a different style border, numbered “Local Guidance n”. Local guidance is provided for customs and conventions that were adopted for a particular usage of this methodology, are not a part of the methodology itself, and are not implemented or enforced by the PD extensions.

1.6.2 Style Choices

These ELDM guidelines also include some specific style standards to further facilitate the sharing of models among those following these guidelines. For example, these guidelines include specifications for font sizes, colors, adornments, and visible properties on diagrams, to list a few.

1.7 Sources

A number of the ideas for this methodology were based on concepts described in the following publications, which can be studied for a deeper understanding of these guidelines:
h. XML in Data Management, Understanding and Applying Them Together by Peter Aiken and M. David Allen.

1.8 Structure (of this document)

There are a number of concepts that need to be understood in order to develop an ELDM. The following are major concepts that get instantiated:

- Model
- Business Area
- Diagram
- Class (or Entity)
- Attribute
- Relationship
- Primitive Data Type
- Domain
- Enumerated List
- Data Structure
- Union.
A separate section is devoted to each of the above concepts. Each concept requires a number of properties to be populated to fully specify an instance of the concept. For example, an instance of a class requires a name, definition, stereotype, etc. There are three kinds of properties:

- Properties that are unique to the concept
- Common properties (i.e., to all of the above concepts)
- Security-related properties.

PD Guidance 2

For each of the above concepts, the PD tool provides a “Concept Properties” Worksheet which can be opened to view and/or edit the properties needed to fully specify the concept. For each concept, there are usually several ways to display the Concept Properties Worksheet. One common way is to find the name of the desired concept object (i.e., the name of the diagram, class, attribute, etc.) in the PD browser, right mouse click on the name, and select “Properties”. A Concept Properties Worksheet, similar to the one below, will appear.
Interspersed throughout this document are instructions for locating properties of various concepts. The instructions may refer to worksheets, tabs, sub-tabs, sections, columns and fields. For example, in the figure above, an instruction might reference the Class Properties worksheet, General tab, Definition section, and Source field.

Note that there is a box in the lower left corner of every properties worksheet, providing the option “More” or “Less”. You should normally select the “Less” option, which will display fewer tabs (and provide the option to select “More”), as shown in the above diagram. If you have selected “Less” and have followed all the recommended setup instructions, the only tabs which will be displayed are those which display one or more properties mentioned in this Data Modeling Guide. Other (undisplayed) tabs are either unused in this implementation, or are used for very advanced, somewhat esoteric, purposes that are beyond the scope of this guide.

Nearly all Concept Worksheets will display the following three tabs when the “Less” option is selected:
General, Version Info, and Requirements. Most Concept Worksheets will display additional tabs that contain properties required by those particular concepts.

APPENDIX A shows the properties that are unique to each concept and shows the interrelationships between the concepts. Appendix B (ELDM Common Properties) shows and defines the common properties. A separate document, the “Enterprise LDM Security Marking Guidelines” defines the security-related properties.

There are additional concepts that are important to understanding an ELDM (e.g., stereotypes, cardinality, etc.); explanations for these concepts are included throughout the document, as appropriate.

The following appendices are included with this document:

- APPENDIX A – ELDM Tool-Independent Metamodel
- APPENDIX B – ELDM Common Properties
- APPENDIX C – ELDM Naming Guidelines
- APPENDIX D – ELDM Acronym Guidelines
- APPENDIX E – ELDM Definition Guidelines
- APPENDIX F – ELDM Definition Sourcing Guidelines
- APPENDIX G – Additional ELDM Guidelines
- APPENDIX H – Overview of ELDM Stereotypes
- APPENDIX I – Modifications to Guizzardi
- APPENDIX J – End Notes.
2 Model Concept

A model is a collection or repository of graphical and textual items intended to represent the information needed by an enterprise.

For an ELDM effort, there will be only one “active” model at any given point in time (i.e., the “ELDM”). There will be multiple versions, however, of this model developed over time, each with a separate version number. Further, the model will consist of multiple “Business Areas”. For example, an ELDM version 1.0 may group together the following “Business Areas”: Party, Requirement, and Report.

Eventually, the model (i.e., a version of an ELDM) will cover all business areas of the enterprise.

2.1 Model Properties

PD Guidance 3

Most of the model properties are found in one of the tabs of the Model Properties worksheet, which can be viewed by double clicking the model in the browser. The appropriate tabs and fields are described later in this section or in APPENDIX B for the “Common Properties”.

Normally, you should view the Model Properties worksheet with your “Favorite tabs” available; in other words, when viewing the Model Properties worksheet, the word “More” should appear in the box in the lower left corner of the worksheet. If “Less” is displayed, click on the box to toggle the setting.

For ELDM work, the following tabs are the recommended favorites for the Model Properties worksheet: General, Also Known As, Requirements, and Version Info. To customize your favorite tabs, click on the Property Sheet Menu button to the right of the More (Less) box and select “Customize Favorite Tabs”. Ensure that each of the recommended favorites is checked. Unless you regularly work with PDMs or Requirements Models, you should uncheck the other options.

Note that the favorite tabs customization does not work as desired for all concepts. Some tabs, such as the Name Encoding and Physical Encoding tabs cannot be effectively removed from your favorites for certain worksheets. They are not needed for ELDM work, but will appear nevertheless. Also, some tabs that are required for ELDM work cannot be effectively added to your favorites for certain worksheets. In those cases, you will have to select More for that worksheet.

Note also that this customization applies to the way models are viewed at the workstation you are currently using. It does not change the model itself.

Figure 2-1, taken from Appendix A (ELDM Tool-Independent Metamodel), depicts the Model concept and its properties and relationships to other concepts:
2.1.1 Common Model Properties

The Model concept has a number of common properties (as indicated by the “{Common}” annotation below the name of the “Model” concept in Figure 2-1. The common properties are shown and defined in APPENDIX B (ELDM Common Properties). One of the common properties for a Model, the “definition”, requires the following additional explanation.

Model Property: Definition

The centralized ELDM management team will provide the definition of the model to reflect what it represents and the role it plays in the enterprise.

2.1.2 Additional Model Properties

The Model concept has the following additional property:

Model Property: Version (mandatory)

The model’s “version” property is a sequential number that, when applied to a model name, reflects the variation of the model compared to previous versions of the model.

Note: For an ELDM, there will be one model name (e.g., “ELDM”) that will contain all of an ELDM. A sequential version number, however, will be applied to the model name in order to keep track of changes that occur to this model over time. Thus, version 1.1 of an ELDM model would be a more recent variation than version 1.0 of an ELDM model.
2.1.3 Model Properties Based on Relationships to other Concepts

The Model concept also has the following property, which is based on a relationship to another concept in the ELDM:

**Model Property: Consists of business areas (mandatory)**

The model’s “consists of business areas” property is a list of all of the children business areas of the model.

PD Guidance 5

To view the list of all children business areas of the model, open the default diagram of the Model. From the Menu Toolbar, select “Model” and click on “Packages...”. A List of Packages window will open, containing, in the Name column, all children business areas of the model.
3 Business Area Concept

A business area is a subset of a model, often focused on one independent class (i.e., the primary focus or subject of the business area) and its dependent classes. A business area focus may indeed reflect a critical operational domain of the enterprise. For example, there might be operational business areas called Manufacturing and Shipping. Alternatively, a business area might focus on a semantic domain of interest of the enterprise. For example, there may be a business area called Party\(^1\) that consists of the independent class Party and its dependent classes, such as Party Name.

Note: A business area may include additional independent classes (along with their respective dependent classes) if there are no other stakeholders within the enterprise willing to take over stewardship for defining these additional classes or sometimes if there is a business function that uses all of these classes in performing that function.

A business area may contain one or more business areas if it aids communication or comprehension. For example, the Party business area may become so complex that it makes sense to create Person as a separate, subordinate business area.

PD Guidance 6

Business areas are implemented as PD “packages”. Thus, in the above example, Party would be implemented as a package and Person would be a sub-package of the Party package.

A business area may consist of diagrams, classes, relationships, and data types. Each of these items will be “owned” by a particular business area, which indicates that it is within the context of this business area that the item has its primary purpose. For example, the Party and Party Name classes may be owned by the Party business area. Although these two classes may exist on diagrams, such as the Report diagram, which is owned by another business area (e.g., the Report business area), these two classes primarily exist to support the Party area of the business.

Note: There will be cases where it is not immediately clear which business area should own a class or relationship. In these cases, the centralized ELDM management team should be consulted, if needed, to assist with selecting the most appropriate business area. For example, it may not be immediately clear which business area should “own” an associative class between Party and Report. One of the following three business areas will need to be selected:

1. Party;

2. Report; or

---

\(^1\) In this guide, when referring to the names of actual business areas, diagrams, classes or attributes, the names will be formatted in bold Calibri font. In addition, the first letter of each word in the name will be capitalized for actual business areas, diagrams, and classes. All of the words in attribute names, however, will be shown in lower case form.
A business area focused on the associative class itself, if it is complex enough (i.e., with multiple classes and relationships) to warrant a separate business area for this purpose.

### 3.1 Business Area Properties

Most of the business area properties are found in one of the tabs of the Package Properties Worksheet, which can be viewed by double clicking the package in the browser corresponding to the business area. The appropriate tabs and fields are described later in this section or in APPENDIX B for the “Common Properties”.

Normally, you should view the Package Properties worksheet with your “Favorite tabs“ available; in other words, when viewing the Package Properties worksheet, the word “More” should appear in the box in the lower left corner of the worksheet. If “Less” is displayed, click on the box to toggle the setting.

For ELDM work, the following tabs are the recommended favorites for the Package Properties worksheet: General, Also Known As, Requirements, Dependencies, and Version Info. To customize your favorite tabs, click on the Property Sheet Menu button to the right of the More {Less} box and select “Customize Favorite Tabs”. Ensure that each of the recommended favorites is checked. Unless you regularly work with PDMs or Requirement Models, you should uncheck the other options.

Note that the favorite tabs customization does not work as desired for all concepts. Some tabs, such as the Name Encoding and Physical Encoding tabs cannot be effectively removed from your favorites for certain worksheets. They are not needed for ELDM work, but will appear nevertheless. Also, some tabs that are required for ELDM work cannot be effectively added to your favorites for certain worksheets. In those cases, you will have to select More for that worksheet.

Note also that this customization applies to the way models are viewed at the workstation you are currently using. It does not change the model itself.

Figure 3-1, taken from Appendix A (ELDM Tool-Independent Metamodel), depicts the Business Area concept and its relationships to other concepts.
3.1.1 Common Business Area Properties

The Business Area concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). One of the common properties for a Business Area, the “name”, requires the following additional explanation.

**Business Area Property: Name**

In general, the business area name should reflect the main independent class in the business area. For example, the **Party** business area has the class **Party** as the main class.

---

**PD Guidance 8**

The “name” property of a Business Area appears in the Package Properties worksheet, General tab, Name field.

---

3.1.2 Business Area Properties Based on Relationships to other Concepts

The Business Area concept also has the following properties, which are based on relationships to other concepts in the ELDM:
**Business Area Property: Part of model (mandatory)**

The business area’s “part of model” property is the parent model of the business area.

**Note:** An ELDM model is the parent model of all business areas contained within it.

PD Guidance 9

The “part of model” property of a business area is the model found in the PD browser above the package representing the business area.

**Business Area Property: Consists of business areas (optional)**

The business area’s “consists of business areas” property is a list of the children business areas of the business area, if appropriate (i.e., if this business area has one or more sub-business areas). For example, as described previously, the Party business area may have Person business area as a child business area.

PD Guidance 10

To view the list of all children business areas of a business area, open the default diagram of the business area package. From the **Menu Toolbar**, select **“Model”** and click on **“Packages...”**. A **List of Packages** window will open, containing, in the **Name** column, all children business areas of the desired business area.

**Business Area Property: Part of business area (optional)**

The business area’s “part of business area” property is the parent business area of the business area, if appropriate (i.e., if this business area is a sub-business area of another one). For example, as described previously, the Person business area may be part of the Party business area.

PD Guidance 11

The “part of business area” property of a business area, if any, can be determined from its placement within the PD browser.

**Business Area Property: Owns diagrams (optional)**

The business area’s “owns diagrams” property is a list of the diagrams that the business area owns (i.e., the diagrams whose primary purpose is to describe the data of the business area). For example, the Party and Party Name diagrams primarily exist to describe the data of the Party business area. Thus, the Party business area owns the Party and Party Name diagrams.

PD Guidance 12

The “owns diagram” property of a business area, if any, can be determined from the placement of the diagrams within the PD browser.
**Business Area Property: Owns classes (optional)**

The business area’s “owns classes” property is a list of the classes that the business area owns (i.e., the classes that are “homed” within the business area; in other words, these classes primarily represent or describe data about the focus class (or classes) of the business area.

PD Guidance 13

To view the list of the classes that the business area owns, open the default diagram of the business area package. From the *Menu Toolbar*, select “Model” and click on “Classes...”. A *List of Classes* window will open, containing, in the *Name* column, all classes of the desired business area. Make sure that the (Include Shortcuts) and (Include Subpackages) icons are not selected.

**Business Area Property: Owns relationships (optional)**

The business area’s “owns relationships” property is a list of the non-shortcut relationships that the business area owns (i.e., the relationships that are “homed” within the business area; in other words, these relationships primarily describe relationships involving a class that is “homed” within the business area).

PD Guidance 14

To view the list of the relationships that the business area owns, open the default diagram of the business area package. From the *Menu Toolbar*, select “Model” and click on “Associations...”. A *List of Associations* window will open. Do the same twice more, clicking on “Specialization Objects...” and “Realization Objects...”. A *List of Generalizations* window will open both times, but their contents will be specializations and realizations, respectively. Together, these three windows contain, in their *Name* columns, all relationships the business area owns. Make sure that the (Include Shortcuts) and (Include Subpackages) icons are not selected on any of the three windows.

**Business Area Property: Owns data types (optional)**

The business area’s “owns data types” property is a list of the data types that the business area owns (i.e., the data types that are “homed” within the business area; in other words, these data types primarily represent or describe data types of the business area).

PD Guidance 15

To view the list of the data types that the business area owns, open the default diagram of the business area package. From the *Menu Toolbar*, select “Model” and click on “Domain Objects...”. A *List of Domain Objects* window will open. Do the same thrice more, clicking on “Enum Objects...”, “Struct Objects”, and “Union Objects...”. Appropriate windows will open each time. Together, these four windows contain, in their *Name* columns, all data types the business area owns. Make sure that the (Include Shortcuts) and (Include Subpackages) icons are not selected on any of the four windows.
4 Diagram Concept

A diagram is a view of a set of classes, attributes, and relationships, developed for a particular purpose. There can be multiple diagrams for each business area.

In general, diagrams should be organized to represent a well-bounded set of data and include relationships between classes that are needed to show how the diagram fits within the ELD. Further, in general, each diagram should contain one “focus” class (i.e., the main class that the data modeler is attempting to describe on the diagram) along with all of its dependent classes. Typically this “focus” class is an “independent” class, but it could be another type of class in the following cases:

- If an independent class has so many dependent classes that if they were all included on one diagram, the diagram would not be readable (as explained below in section 4.3, “Other Diagram Guidelines”), then multiple diagrams may be developed (e.g., separate diagram(s) with a dependent class as the “focus” class on each one) to ensure that each diagram is readable.

- A separate diagram may be developed for metadata that has broad applicability throughout the model, even if there is not an “independent” class on the diagram. APPENDIX G (Additional ELD Guidelines) includes a “Broad Application of Metadata” pattern that describes how to deal with this issue.

4.1 Diagram Properties

Most of the diagram properties are found in one of the tabs of the Class Diagram Properties worksheet, which can be viewed by right clicking the (diagram) icon in the browser and selecting “Properties”. The appropriate tabs and fields are described later in this section or in APPENDIX B for the “Common Properties”.

Special Note for Class Diagram Properties Worksheets: These worksheets contain both a General Tab and an Extended General Tab. Please use the Extended General Tab, not the General Tab, which is an undesired vestige that cannot be removed. Also note that on the Extended General Tab, the Definition Field can be directly edited. This is different from the General Tab for all other concepts, where the “Edit” button must be employed to edit the Definition Field.

Normally, you should view the Diagram Properties worksheet with your “Favorite tabs” available; in other words, when viewing the Diagram Properties Worksheet, the word “More” should appear in the box in the lower left corner of the worksheet. If “Less” is displayed, click on the box to toggle the setting.

For ELD work, the following tabs are the recommended favorites for the Class Diagram Properties Worksheet: General, Extended General, Also Known As, Requirements, Dependencies, and Version Info. To customize your favorite tabs, click on the Property Sheet Menu button to the right of the More (Less) box and select “Customize Favorite Tabs”. Ensure that each of the recommended favorites is checked.
Unless you regularly work with PDMs or Requirement Models, you should uncheck the other options.

Note that the favorite tabs customization does not work as desired for all concepts. Some tabs, such as the Name Encoding and Physical Encoding tabs cannot be effectively removed from your favorites for certain worksheets. They are not needed for ELDM work, but will appear nevertheless. Also, some tabs that are required for ELDM work cannot be effectively added to your favorites for certain worksheets. In those cases, you will have to select More for that worksheet.

Note also that this customization applies to the way models are viewed at the workstation you are currently using. It does not change the model itself...

Figure 4-1 below, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Diagram concept and its relationships to other concepts.

![Diagram Concept and Its Relationships](image_url)

### 4.1.1 Common Diagram Properties

The Diagram concept has a number of common properties as shown and defined in Appendix B (ELDM Common Properties). One of the common properties for a Diagram, the “name”, requires the following additional explanation.
Diagram Property: Name

The name of the diagram should reflect the focus class or classes of the diagram (e.g., an independent class, a dependent class, or a set of dependent classes). For example, if the diagram is focused on the dependent class Order Line Item along with classes that help to define the Order Line Item, then “Order Line Item” would be an appropriate name for the diagram.

PD Guidance 17
The “name” property of a diagram appears in the Class Diagram Properties worksheet, Extended General tab, Name field.

4.1.2 Additional Diagram Property

Diagram Property: Diagram Security Banner (optional)

The diagram’s security banner is a text string containing the classification of the information disclosed on the diagram itself.

Local Guidance 2
Note that while the Diagram Security Banner is optional in the methodology, it is required by local guidance.

Also note that the diagram security classification may be less than the classification of the concepts displayed on the diagram, since their classification relates to information shown on concept property worksheets which are not displayed on the diagram.

PD Guidance 18
To place a Security Banner on a diagram, open the diagram and right mouse click on any open space in the diagram. A Diagram Security Banner window will open, containing a list of commonly used classifications. Select the desired classification and click on the “OK” button. A pair of properly placed banners will appear on the diagram.

Note: If the desired classification is not in the list, select the nearest classification to the desired one. After the banners are placed, you may edit each banner to display the correct classification.

PD Guidance 19
Note: The Diagram Property worksheet, Other section contains a Diagram Security Banner field which displays the classification selected from the window described above. It does not reflect any edited changes made to the banner on the diagram. Nor is changing this field reflected back on the diagram. Do not use this field to adjust the Diagram Security Banner.
4.1.3 Diagram Properties Based on Relationships to other Concepts

The Diagram concept also has the following properties, which are based on relationships to other concepts in the ELDM:

*Diagram Property: Owned by business area (mandatory)*

The diagram’s “owned by business area” property is the business area that owns the diagram (i.e., the business area within which the diagram has its primary purpose).

**PD Guidance 20**
- The “owned by business area” property of a diagram is found in the *Class Diagram Properties* worksheet, *Extended General* tab, *Other* section, *Parent* field.

*Diagram Property: Contains classes (mandatory)*

The diagram’s “contains classes” property is a list of the classes that appear on the diagram.

**PD Guidance 21**
- The “contains classes” property of a diagram consists of all items labeled “Class” found in the *Class Diagram Properties* worksheet, *Dependencies* tab, *Short Description* column.

*Diagram Property: Contains data structures (optional)*

The diagram’s “contains data structures” property is a list of the data structures that appear on the diagram.

**PD Guidance 22**
- The “contains data structures” property of a diagram consists of all items labeled “Struct” found in the *Class Diagram Properties* worksheet, *Dependencies* tab, *Short Description* column.

*Diagram Property: Contains relationships (optional)*

The diagram’s “contains relationships” property is a list of the relationships that appear on the diagram.

**PD Guidance 23**
- The “contains relationships” property of a diagram consists of all items labeled “Association,” “Specialization,” or “Realization” found in the *Class Diagram Properties* worksheet, *Dependencies* tab, *Short Description* column.

*Diagram Property: Contains enumerated lists (optional)*

The diagram’s “contains enumerated lists” property is a list of the enumerated lists that appear on the diagram.
The “contains enumerated lists” property of a diagram consists of all items labeled “Enum” found in the Class Diagram Properties worksheet, Dependencies tab, Short Description column.

**Diagram Property: Contains domains (optional)**

The diagram’s “contains domains” property is a list of the domains that appear on the diagram.

The “contains domains” property of a diagram consists of all items labeled “Domain” found in the Class Diagram Properties worksheet, Dependencies tab, Short Description column.

**Diagram Property: Contains unions (optional)**

The diagram’s “contains unions” property is a list of the unions that appear on the diagram.

The “contains unions” property of a diagram consists of all items labeled “Union” found in the Class Diagram Properties worksheet, Dependencies tab, Short Description column.

### 4.2 Types of Diagrams

Each business area should at least have a “main” diagram and may have “additional” diagrams, depending upon the complexity of the business area. Further, for those business areas with two or more diagrams, an “overall” diagram may be constructed, as described below. Thus, there are three categories of diagrams:

1. “Main” diagram of a business area:
   
   a. One diagram for each business area should have the same name as the business area itself. (If the business area also contains an “overall” diagram, the word “Main” should be prepended to the business area name to form the “main” diagram name.) It should contain the “focus” class of the business area, which is typically the primary independent class (e.g., **Party**) and it should contain as many as possible of the following categories of classes related to this class, as appropriate, staying within the “readable” guideline explained below:
      
      - Entities involved in “specialization” relationships with the focus class
      - “Dependent” classes of the focus class (e.g., **Party Name**)  
      - “Associative” classes related to the focus class (and possibly other classes on which the associative class depends for its existence, if needed for clear communication)
      - Other classes directly related to the focus class.
2. “Additional” diagrams of a business area:
   - A business area may have additional diagrams, as needed, to describe the related classes that did not fit on the main diagram. These diagrams should be named to reflect the name of the “focus” of that diagram, either a particular “focus” class on that diagram or a descriptive term relating the classes on that diagram. Where possible, the classes that are selected to be on the same diagram should have some affinity or likeness with the other classes selected (e.g., they all help to describe some aspect of the focus class via relationships to it).

3. Overall diagram of a business area:
   - If a business area has two or more diagrams, an “Overall” diagram (or set of diagrams, if having multiple overall diagrams facilitates communication) may be created as a mechanism to communicate the full breadth of the focus classes (i.e., a “360 degree view” of the focus class or classes). This overall diagram should have the same name as the business area with the word “Overall” prepended (or one may devise another appropriate naming convention if two or more diagrams are required to fully encompass the entire business area).

The “Overall” diagram(s) should contain:

a) The focus class (or classes) of the business area (typically in or near the center of the diagram).

b) For each focus class:
   - Include all classes “one-out” (i.e., all classes of any stereotype that are directly related to the focus class via any kind of relationship).
   - If that “one-out” class is a superclass of the focus class via a specialization or realization relationship, include each of the superclass’s parent classes via specialization or realization relationships all the way to the top or until the superclass is the focus of another business area, whichever comes first.

c) For each of the other “one-out” classes that are not the focus of a sub-business area diagram:
   - If that “one-out” class has a non-dependency relationship to it, no additional classes related to the “one-out” class need to be shown on the diagram.
- If that “one-out” class is a subclass <<role>> class and an <<associative>> class exists to describe how the <<role>> class is used or how it was established, optionally include the <<associative>> and its other parent(s) (i.e., another <<role>> or a <<kind>> class), especially if the <<associative>> is not owned by another diagram.

- If that “one-out” class is a <<role category>> class and an <<associative>> class exists to describe how the <<role category>> class is used or how it was established, include the <<associative>> and its other parent(s) (i.e., <<role>> or <<kind>> class).

- If that “one-out” class is a <<dependent>> class, include all of the one-out classes from that <<dependent>> class (and all further levels of <<dependent>> classes of the <<dependent>> class, as far as they exist).

- If that “one-out” class is an <<associative>> class, include the other parent class of that <<associative>> class and any other “one-out” classes from the <<associative>> class.

- If that “one-out” class is a subclass, include all of the subclass’s one-out classes and for each one, repeat step c above.

Figure 4-2 below graphically depicts the above rules for the “Overall” diagram, in UML notation.

![Figure 4-2 Example of Overall Diagram Rules](image-url)
4.3 Other Diagram Guidelines

In general, a diagram should be readable when it is printed on 11” x 17” size paper. An exception to this general rule would likely be needed for “overall” diagrams. As needed, a diagram should be printed or plotted on a larger size sheet of paper in order to be readable.

Where possible, each child class involved in a “specialization” relationship (see sections 5.1.2.1.6 and 7.1.2.1.1.2 later in this document) should be shown below its superclass.

Relationship lines should not cross, where possible.

A relationship line should never be routed behind (or in front of) a class or note.

In general, the names of all attributes for a class should be displayed (i.e., made visible) on diagrams. However, for some audiences, it may be appropriate to only show the class name (i.e., without displaying attribute names).

In general, the order of attributes shown within a class should be as follows:

- Partitioning attribute(s)
- Candidate key(s), i.e., attribute(s) which might serve as primary identifier(s) in a PDM
- Other mandatory attribute(s)
- Remaining attribute(s), semantically grouped, where appropriate (e.g., begin date, end date) or listed alphabetically.

Each diagram should contain a label that includes the model name, business area name, and diagram name.

<table>
<thead>
<tr>
<th>PD Guidance 27</th>
</tr>
</thead>
<tbody>
<tr>
<td>To insert a label on a diagram, select the (Title) Icon in the Palette toolbar, and click on the diagram. A properly filled-in diagram label will appear.</td>
</tr>
</tbody>
</table>

Table 4-1 below shows, for each of the items that may be placed on a diagram, where the item should be placed on the diagram, the appropriate background color, the appropriate text color, and the appropriate font that should be used.
Table 4-1 Placement, Color and Font for Items on Diagrams

<table>
<thead>
<tr>
<th>Item on Diagram</th>
<th>Placement on Diagram</th>
<th>Background Color</th>
<th>Text Color</th>
<th>Text Font</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall Security Label</td>
<td>Left justified at both the top left and bottom left of the diagram</td>
<td>Depends on security label level (see separate Security Marking Guidelines)</td>
<td>Yellow</td>
<td>Arial Regular 18</td>
</tr>
<tr>
<td>Diagram Label</td>
<td>At the top, to the right of the security label</td>
<td>White</td>
<td>Black</td>
<td>Lucida Console (which is a fixed-width font), Regular 16</td>
</tr>
<tr>
<td>Classes owned by business area of diagram</td>
<td>As appropriate</td>
<td>Yellow</td>
<td>Black (i.e., for all visible text within class)</td>
<td>Arial Regular 12 (i.e., for all visible text within class)</td>
</tr>
<tr>
<td>Classes owned by business area other than the one of the diagram</td>
<td>As appropriate</td>
<td>Light Blue</td>
<td>Black (i.e., for all visible text within class)</td>
<td>Arial Regular 12 (i.e., for all visible text within class)</td>
</tr>
<tr>
<td>Enumerated Lists, Data Structures, Domains, and Unions owned by business area of diagram</td>
<td>As close to attribute as practically possible</td>
<td>Light Green</td>
<td>Black</td>
<td>Arial Regular 12</td>
</tr>
<tr>
<td>Enumerated Lists, Data Structures, Domains, and Unions owned by business area other than the one of the diagram</td>
<td>As close to attribute as practically possible</td>
<td>Pale Red</td>
<td>Black</td>
<td>Arial Regular 12</td>
</tr>
<tr>
<td>Relationship names</td>
<td>As close to the Relationship line as possible, ensuring that it cannot be mistaken as the name of another Relationship line</td>
<td>N/A</td>
<td>Black</td>
<td>Arial Regular 12</td>
</tr>
<tr>
<td>Relationship and outline lines</td>
<td>As appropriate</td>
<td>Black</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Notes</td>
<td>As close to concept being explained as possible</td>
<td>Purple</td>
<td>Black</td>
<td>Arial Regular 12</td>
</tr>
</tbody>
</table>

Table 4-2 below shows the appropriate Red-Green-Blue (RGB) settings that apply to each of the colors in the above table.
Table 4-2 RGB Settings for Colors

<table>
<thead>
<tr>
<th>Color</th>
<th>RGB values</th>
<th>Item on a Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>0, 0, 0</td>
<td>Text, relationship lines, and borders of classes, structures or text boxes.</td>
</tr>
<tr>
<td>Light Blue</td>
<td>202,255,255</td>
<td>Background of classes owned by business area other than the one of the diagram</td>
</tr>
<tr>
<td>Light Green</td>
<td>207,255,207</td>
<td>Background of enumerated lists and data structures owned by business area of diagram</td>
</tr>
<tr>
<td>Pale Red</td>
<td>255,207,207</td>
<td>Background of enumerated lists and data structures owned by business area other than the one of the diagram</td>
</tr>
<tr>
<td>Purple</td>
<td>209, 209, 233</td>
<td>Background of notes</td>
</tr>
<tr>
<td>White</td>
<td>255, 255, 255</td>
<td>Background of diagram labels</td>
</tr>
<tr>
<td>Yellow</td>
<td>255, 255, 191</td>
<td>Background of classes owned by business area of the diagram</td>
</tr>
</tbody>
</table>

A feature has been added to the PowerDesigner extensions to allow developers to more easily track changes to models. Newly created packages are considered “unapproved”. When a package in a model is considered complete, a flag can be set to “approve” the package. Until that time, the color settings for the display of the certain concepts belonging to the package are different from those in Table 4-1 and Table 4-2. The different color settings are in effect for those concepts on all diagrams, whether or not they belong to the package.

Prior to package approval, there are three exceptions:

1. Classes and Data Types have a light blue shadow.
2. Relationship lines appear as thick light blue, rather than standard black.
3. Attribute names are light blue rather than black.

When the package is approved, (and the Set Colors script, documented elsewhere, is invoked) classes, relationships and attributes owned by the package are displayed in standard colors on all diagrams (whether or not the diagram is owned by the package).

As modeling continues, any new, or newly edited, classes, attributes and relationships will appear in the exception (unapproved) colors, until the package to which they belong is approved.
5 Class (or Entity) Concept

Throughout this document the term “class” will be used to refer to the ERD concept of an “entity” or the UML concept of a “class”.

A class is a set of like persons, places, things, events, or concepts about which data are kept. Every class:

- Must represent something about which the enterprise wishes to keep data¹
- Must not represent a particular thing (i.e., instance), but rather a set of like or similar things²
- Except for events, must have a principle of identity. (See section 5.1.2.1.3 below).

5.1 Class Properties

Figure 5-1 below, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Class concept and its properties and relationships to other concepts.

PD Guidance 29

Most of the Class Properties are found in one of the tabs of the Class Properties Worksheet, which can be viewed by double clicking the class in the browser or on a diagram. The appropriate tabs and fields are described later in this section or in Appendix B for the “Common Properties”.

Normally, you should view the Class Properties Worksheet with your “Favorite tabs” available; in other words, when viewing the Class Properties Worksheet, the word “More” should appear in the box in the lower left corner of the worksheet. If “Less” is displayed, click on the box to toggle the setting.

For ELDN work, the following tabs are the recommended favorites for the Class Properties Worksheet: General, Detail, Attributes, Also Known As, Rules, Requirements, Dependencies, and Version Info. To customize your favorite tabs, click on the Property Sheet Menu button to the right of the More (Less) box and select “Customize Favorite Tabs”. Ensure that each of the recommended favorites is checked. Unless you regularly work with PDMs or Requirement Models, you should uncheck the other options.

Note that the favorite tabs customization does not work as desired for all concepts. Some tabs, such as the Name Encoding and Physical Encoding tabs cannot be effectively removed from your favorites for certain worksheets. They are not needed for ELDN work, but will appear nevertheless. Also, some tabs that are required for ELDN work cannot be effectively added to your favorites for certain worksheets. In those cases, you will have to select More for that worksheet.

Note also that this customization applies to the way models are viewed at the workstation you are currently using. It does not change the model itself.
5.1.1 Common Class Properties

The Class concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). Two of the common properties for a class, the “name” and “definition”, require the following additional explanation.

**Class Property: Name**

The class name must comply with the ELDM Naming Guidelines (in Appendix C) and the ELDM Acronym Guidelines (in APPENDIX D).

PD Guidance 30
The “name” property of a class appears in the Class Properties worksheet, General tab, Name field.

**Class Property: Definition**

The class definition must conform to ELDM Definition Guidelines (Appendix E).
5.1.2 Additional Class Properties

The Class concept has the following additional properties:

**Class Property: Stereotype (mandatory)**

The class’s “stereotype” property is used to indicate that the class conforms to a certain set of characteristics, described in full below, with supplementary information provided in Appendix H (Overview of ELDM Stereotypes). These stereotypes are based on a work by Guizzardi. The “stereotype” name is placed above the class name on a diagram. All classes in an ELDM will be marked as being one of the following seven stereotypes:

1. Kind
2. Role
3. Category
4. Role Category
5. Dependent
6. Associative
7. Event.

To add a class to a diagram, the modeler selects the icon which matches the desired stereotype for the class and clicks on the appropriate diagram to place the class. The class’s “Stereotype” property is set automatically based on the selected icon. The following class icons are available in the Base Extension toolbar of PD:
Event

An existing class’s “Stereotype” property can be found in the Class Properties worksheet, General tab, Stereotype field, and can be modified by selecting a new stereotype from the pull down list.

On a diagram, the class stereotype appears within the class rectangle, above the class name, bracketed by the symbols << and >>.

Class Property: Is abstract (mandatory)

The class’s “is abstract” property is a Boolean indicator of whether the class is abstract (true) or concrete (false). See section 5.1.2.1.4 below for a discussion of abstract and concrete classes.

PD Guidance 33

The class’s abstract property is indicated on the Class Properties worksheet, General tab, Abstract checkbox. PD automatically sets the box for Categories and Role Categories, which are the only class stereotypes which should be abstract in the logical model.

5.1.2.1 Class Principles

- Before discussing the details of the class stereotypes, it is essential to understand several principles that are used in this methodology for conceptual modeling to identify the class stereotypes. The seven stereotypes (Event, Kind, Category, Role, Role Category, Dependent, and Associative) will be defined in terms of one or more of the following dichotomies: continuant/occurrent [section 5.1.2.1.1], dependent/independent [section 5.1.2.1.2], concrete/abstract (unified/disparate principle of identity) [section 5.1.2.1.4], and permanent/transient [section 5.1.2.1.7].

- It must be understood that these principles are always defined with respect to the universe (the business) being modeled.

- Table 5-1 below shows the principles used to define the seven class stereotypes.

Table 5-1 Class Defining Principles

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Continuant/Occurrent</th>
<th>Dependent/Independent</th>
<th>Concrete/Abstract</th>
<th>Permanent/Transient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Continuant</td>
<td>Independent</td>
<td>Concrete</td>
<td>Permanent</td>
</tr>
<tr>
<td>Category</td>
<td>Continuant</td>
<td>Independent</td>
<td>Abstract</td>
<td>Permanent</td>
</tr>
<tr>
<td>Role</td>
<td>Continuant</td>
<td>Independent</td>
<td>Concrete</td>
<td>Transient</td>
</tr>
<tr>
<td>Role Category</td>
<td>Continuant</td>
<td>Independent</td>
<td>Abstract</td>
<td>Transient</td>
</tr>
<tr>
<td>Dependent</td>
<td>Continuant</td>
<td>Dependent (single)</td>
<td>Concrete</td>
<td>Permanent</td>
</tr>
</tbody>
</table>
### 5.1.2.1.1 Continuant and Occurrent Classes

Many classes represent things, or objects, more technically known as **continuants**. Although instances of continuants may change over time, they fully exist (and can be fully described) at any point in time. We say continuants exist “in time.”

Other classes represent events, more technically known as **occurrents**. An occurrent happens over time, and cannot be fully described until it is complete. To fully describe an occurrent, one must describe its change over time. We say occurrents are “of time.”

### 5.1.2.1.2 Independent and Dependent Classes

A class is said to be **existentially dependent** on another class if and only if any instance of the first class depends on the existence of an instance of the second class both to come into existence and to remain in existence. Of the following three bullets, only the third is an example of existential dependence.

- An instance of a person depends on the existence of its mother to come into existence, but does not depend upon the existence of its mother to remain in existence
- A typical parasite may depend on the existence of its host to remain in existence, but did not depend on the existence of its host to come into existence
- A particular skill of a particular person depends on the existence of that particular person both to come into existence and to remain in existence.

A class is said to be **independent** if and only if it is not existentially dependent on another class. Both **Person** and **Parasite** are classes which are independent. In this methodology, independence is synonymous with existential independence.

A class is said to be **dependent** if and only if it is existentially dependent on at least one other class. **Person Skill** is a class which is dependent. In this methodology, dependence is synonymous with existential dependence.

It will be seen later that two class stereotypes, Dependent and Associative, will be defined in part by the dependent determinant, while the other four continuant stereotypes, Kind, Role, Category, and Role Category, will be defined in part by the independent determinant.

---

**Table:**

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Continuant/Occurrent</th>
<th>Dependent/Independent</th>
<th>Concrete/Abstract</th>
<th>Permanent/Transient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associative</td>
<td>Continuant</td>
<td>Dependent (multiple)</td>
<td>Concrete</td>
<td>Permanent</td>
</tr>
<tr>
<td>Event</td>
<td>Occurrent</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

---
5.1.2.1.3 Principle of Identity of a Class

The enterprise model concerns itself with “identity,” while a physical model or database concerns itself with “identifiers”. Identity can be defined as the fact of being who or what a person or thing is. An identifier is the label by which we distinguish or communicate the uniqueness (the identity) of something.

A principle of identity is defined as the method(s) or rule(s) by which we judge if two (apparent) instances of some concept are identically the same instance of the thing. In other words, a principle of identity is the method(s) or rule(s) by which we distinguish two different instances of a class.

Modelers must agree on the principle of identity of a concept, but there is no requirement to document the principle of identity. Often, the principle is arrived at by common sense; sometimes, it is the catalyst for great debate.

5.1.2.1.4 Concrete and Abstract Classes

A class is said to be concrete if and only if it can be directly instantiated, that is, it can be directly represented by an actual example of the concept represented by the class. The principle of identity of the instance must be that of the class. For example, if we define the class Person to represent human beings, then any individual human could serve as an instantiation of the class. The principle of identity that applies to the individual human is the same principle of identity that applies to the class. Person is therefore an example of a concrete class.

A class is said to have a Unified Principle of Identity if and only if there exists some common way which can distinguish one instance of the class from another. The concepts of a class being concrete and a class a unified principle of identity are equivalent.

A class is said to be abstract if and only if it cannot be directly instantiated. For example, if we define the class Party to be either a human being (a member of the class Person) or an organized grouping of human beings (a member of the class Organization), then an individual human being or a specific social or business grouping could serve as an instantiation of the class. However the principle of identity that applies to human beings is not the same as the principle of identity that applies to social or business groupings, and thus not the same as the (more complex) principle of identity that applies to the class Party. The class Party can only be indirectly instantiated by either a member of the class Person or by a member of the class Organization. Party is therefore an example of an abstract class. Often an abstract class is actually defined as one whose instances belong to one of several sub-classes, just as Party was defined.

A class is said to have a Disparate Principle of Identity if and only if at least two ways are required to distinguish one instance from another. The concepts of a class being abstract and a class having a disparate principle of identity are equivalent.

The distinction that a class will be considered abstract often seems very obvious and clear-cut, but also can seem rather arbitrary, and simply a matter of perspective. In one circumstance or business model, a
certain class may be considered concrete while in another circumstance or business model, that same class might be considered abstract. The modelers must agree, based on the context addressed by their particular model.

- The Concise Oxford English Dictionary defines “appliance” as “a device designed to perform a specific task.” Even though the class is well-defined, its instances are so diverse (toothpick, toaster, refrigerator) that one would likely devise several layers of sub-classification before actually being prepared to discuss direct instantiation. In many circumstances, Appliance would be a good candidate to declare an abstract class.

- The Concise Oxford English Dictionary says that a “refrigerator” is “an appliance or compartment which is artificially kept cool and used to store food and drink.” While there are many ways to classify a Refrigerator, such as by manufacturer, size, color, energy source, or location, one would be hard-pressed to make the argument that Refrigerator is an abstract form of Amana Refrigerator, Electric Refrigerator, or Home Refrigerator. Refrigerator is probably a good example of a concrete class.

- In the taxonomy of living things, there might be great variability among models in deciding a level where abstract ends and concrete begins. Is Living Thing itself abstract or concrete? Carnivores might decide that Animal is a good concrete class; instantiate an Animal and have a good meal. To bird-watchers, Bird might be abstract; they only want to distinguish instances of American Goldfinch or Ruby Throated Hummingbird.

### 5.1.2.1.5 Realization Classes

When an abstract class “X” can be indirectly instantiated by a member of the concrete class “Y”, we say that Y is a realization class of X. In other words, realization represents the taking of something abstract and "making it real.” In the example discussed in the first two paragraphs of section 5.1.2.1.4, both Person and Organization are realization classes of Party.

More precisely, a class (the subclass) is said to be a realization class of another class (the superclass) if and only if:

- The subclass is concrete
- The superclass is abstract
- The principle of identity of the subclass serves as one of the methods comprising the superclass’s disparate principle of identity
- Every instance of the subclass is also an instance of the superclass,
- Not every instance of the superclass is also an instance of the subclass.

We say that the superclass of a realization class is a categorization class.
5.1.2.1.6 Specialization Classes

A class (the child, or subclass) is said to be a **specialization class** of another class (the parent or superclass) if and only if:

- The principle of identity of the subclass is the same as the principle of identity of the superclass
- Every instance of the subclass is also an instance of the superclass
- Not every instance of the superclass is also an instance of the subclass.

Both classes **Man** and **Nurse** can be subclasses of the class **Person** in an enterprise model because modelers could reasonably conclude that all three classes have the same principle of identity, every instance of **Man** or **Nurse** is also an instance of **Person**, but not every instance of **Person** is an instance of **Man** or of **Nurse**.

It will be understood later that a corollary of the above definition is that a subclass has at least one attribute or relationship that the superclass does not have.

We say that a superclass of a specialization is a **generalization class**.

5.1.2.1.7 Permanent and Transient Classes

A class is said to be **transient** if and only if:

- Either
  - It is a subclass of another class
  - An instance of the superclass may be a member of the subclass at some time(s) but need not be an instance of the subclass at all times.
- Or
  - It is a superclass of a transient class via a realization relationship.

A class is said to be **permanent** if and only if it is not transient.

The class **Person** might be modeled in an enterprise model as not being a subclass of another class, and would then be permanent. The class **Man** is a subclass of **Person**, and every instance of **Person** which is ever a **Man** must always be a **Man**, so the class **Man** is permanent. The class **Nurse** is a subclass of **Person**, but an instance of **Person** can sometimes be a **Nurse** while the same instance of **Person** might not always be a **Nurse**. Thus **Nurse** is a transient class.

5.1.2.2 Class Stereotype Details

5.1.2.2.1 Kind

A **kind** class is a continuant that is independent, concrete, and permanent.
How to model a “kind” class:

- The stereotype <<kind>> will be used to mark a “kind” class.

PD Guidance 34
Select the (kind) icon from the Base Extension toolbar of PD and click in the diagram in which the class should appear.

Figure 5-2 below shows an example of Person, a <<kind>> class. Person is a “kind” class for the following reasons:

- An instance of the class Person could “live alone” (i.e., without any other class existing, which makes Person an independent class)
- All instances of Person are direct instantiations; and thus Person is concrete
- The concept of Person is a permanent concept.

```
<<kind>>
Person
name : String
```

Figure 5-2 Example of a Kind Class - Person

5.1.2.2.2 Role
A role class is a continuant that is independent, concrete, and transient.

Additional characteristic of a “role” class:

- Instances are established or used by another <<kind>> or <<role>> class (to reflect the reason the business cares about the role).

How to model a “role” class:

- The stereotype <<role>> will be used to mark a “role” class.

PD Guidance 35
Select the (role) icon from the Base Extension toolbar of PD and click in the diagram in which the class should appear.

- A <<role>> class is to be modeled as a specialization subclass of a <<kind>> class or as a specialization subclass of another <<role>> class. Note: if the <<role>> class is modeled as a subclass of another <<role>> class, there must be a <<kind>> class as the ultimate superclass class of this series of <<role>> classes.
A <<role>> class must have a direct or indirect, mandatory relationship to the <<kind>> or <<role>> class by which the <<role>> was established or used. This relationship is said to be “direct” if it is drawn from this specific <<role>> class; and is said to be “indirect” if it is drawn from a <<role category>> class that is acting as the parent of the <<role>> class via a realization relationship or from another <<role>> class that is acting as either the parent or a child of this <<role>> class.

**Note:** The relationship is to be mandatory because an ELDM is to reflect the data that could exist if the business had “perfect knowledge”, as discussed more fully as a pattern in APPENDIX G (Additional ELDM Guidelines). Further, if the mandatory relationship requires one or more attributes or relationships to fully describe it, this relationship should be drawn through an associative class that includes these attributes or relationships, as depicted in Figure 5-3 below. Also, the name of the associative class should be based on the interaction of the two different classes that the associative class brings together.

---

**Local Guidance 3**

For the local implementation, it has been decided that the <<kind>> or <<role>> class, by which the <<role>> was established or used, need not always be included in the model; hence the relationship need not always be included in the model or drawn.

---

Figure 5-3 shows an example of the **Doctor** <<role>> class, which is a subclass of **Person**. The role of doctor for a person represents a transient set of properties that apply to a person (i.e., a person is not born a doctor, but may become one at a certain point in time and then may stop being a doctor at a later point in time); and the doctor role exists, because there is a license, granted by some licensing authority, which authorizes the doctor to practice his or her profession.

**Figure 5-3 Example of a Role Class - Doctor**
When defining a role class, the definition should explain what caused the role to exist. For example, the **Doctor** role class might be defined as “A person who has received a license to practice in the healing arts.”

5.1.2.2.3 **Category**

A category class is a continuant that is independent, abstract, and permanent.

The realization classes of a category class are mutually exclusive (i.e., no instance of the category can be an instance of more than one of the realization classes).

How to model a “category” class:

- The stereotype <<category>> will be used to mark a “category” class.

  PD Guidance 36
  Select the **C** (category) icon from the Base Extension toolbar of PD and click in the diagram in which the category should appear.

  - The relationships between a category class and its realization classes must be via “realization” relationships (which will be discussed later in section 7.1.2.1.1.1 on page 64).

  - The relationships between a category class and its specialization classes must be via “specialization” relationships (which will be discussed later in section 7.1.2.1.1.2 on page 66).

For example, Figure 5-4 shows the class **Party**, a <<category>> class with two realization classes.
Party is a “Category” class for the following reasons:

- An instance of the class **Party** could “live alone” (i.e., without any other class existing, which makes **Party** an independent class).

- A person is always a person and an organization is always an organization (indicating that both realization classes are “permanent” concepts).

- The class **Party** has disparate subclasses: **Person** and **Organization**. People could be uniquely identified in a number of ways (e.g., retinal scan, fingerprint, DNA, etc.), however, none of these ways could be used to identify an organization.

### 5.1.2.2.4 Role Category

A **role category** class is a continuant that is independent, abstract, and transient.

The realization classes of a role category class are mutually exclusive (i.e., no instance of the role category can be an instance of more than one of the realization classes).

How to model a “role category” class:

- The stereotype **<<role category>>** will be used to mark a “role category” class.

Select the **RC (role category)** icon from the **Base Extension** toolbar of PD and click in the diagram in which the class should appear.
• A role category class must have at least two specialization and/or realization classes. The realization classes must be <<role>> classes, while the specialization classes must be <<role category>> classes.

• The relationships between the <<role category>> class and its realizations must be via “realization” relationships (which will be discussed later in section 7.1.2.1.1.1 on page 64).

• The relationship between the <<role category>> class and its specializations must be via “specialization” relationships (which will be discussed later in section 7.1.2.1.1.2 on page 66).

• The <<role category>> class must have a direct or indirect, mandatory relationship to the <<kind>> or <<role>> class by which the <<role category>> was established or used. This relationship is said to be “direct” if it is drawn from this specific <<role category>> class; and is said to be “indirect” if a mandatory relationship is drawn from each <<role>> class that is acting as a child of this <<role category>> class via a realization relationship or from another <<role category>> class acting as the parent or a child of this <<role category>> class.

Note: These relationships are to be mandatory because an ELDM is to reflect the data that could exist if the business had “perfect knowledge”, as discussed more fully as a pattern in APPENDIX G (Additional ELDM Guidelines). Further, if any of the mandatory relationships require one or more attributes or relationships to fully describe it, this relationship should be drawn through an associative class that includes these attributes or relationships, as depicted in Figure 5-5.

For example, Figure 5-5 shows the class **Customer**, a <<role category>> class.
Customer is a “Role Category” class for the following reasons:

- An instance of the class Customer could “live alone” (i.e., without any other class existing, which makes Customer an independent class).

- The role of Customer Person represents a transient set of properties that apply to a person (i.e., a person is not born a customer, but becomes a customer at a certain point in time and then may cease being a customer at a later point in time) and, similarly, the role of Customer Organization represents a transient set of properties that apply to an organization.

- The class Customer has disparate subclasses: Customer Person and Customer Organization. A Customer Person would be uniquely identified in the way that a person would be identified (e.g., retinal scan, fingerprint, DNA, etc.), however, none of these ways could be used to identify a Customer Organization.

### 5.1.2.2.5 Dependent

A dependent class is a continuant that is existentially dependent on exactly one other class, and is concrete and permanent.

How to model a “dependent” class:
• The stereotype <<dependent>> will be used to mark a “dependent” class.

Select the \( \text{dp} \) (dependent) icon from the Base Extension toolbar of PD and click in the diagram in which the class should appear.

For example, Figure 5-6 shows the class **Person Name**, a <<dependent>> class.

![Diagram of Person Name class](image)

**Figure 5-6 Example of a Dependent Class – Person Name**

**Person Name** is a <<dependent>> class because it depends on **Person** for its existence (i.e., an instance of **Person Name** could not be created without an instance of **Person** first being created). The attributes of **Person Name** describe **Person**.

You might think of dependents often as something that otherwise might be represented as an attribute except that the attribute-like concept itself has multiplicity, attributes or associations that must be represented—i.e., an "attribute on steroids".

### 5.1.2.2.6 Associative

An **associative** class is a continuant that is existentially dependent on one or more other classes as expressed at least twice via dependency associations, has a unified principle of identity, and is permanent. Typically, this means the associative class existentially depends on two (or more) different classes for its existence, but it is possible for an associative to have two dependency associations with one class, or a single dependency association, with a minimum cardinality of 2, with one class.

You might think of associatives as complex associations that require resolution of multiplicities or the addition of attributes or more than two associations.

Additional characteristic of an “associative” class:

• Associative classes often are related to events. In fact, events often represent the continuing state of a relationship represented by an associative, or a change in state that begins or terminates a relationship represented by an associative. So a modeler should, in general, consider the creation of one or more event classes when an associative class is created, and vice versa.

• How to model an “associative” class.
• The stereotype <<associative>> will be used to mark an “associative” class.

Select the (associative) icon from the Base Extension toolbar of PD and click in the diagram in which the class should appear.

For example, Figure 5-7 shows the class Report Person Usage, an <<associative>> class.

Report Person Usage is an <<associative>> class because it depends on both the Report and Person classes for its existence (i.e., an instance of Report Person Usage could not be created without an instance of Person and Report created first).

If the Report Person Usage class did not exist, there would be a need to have a “many-to-many” relationship between a Person and a Report (i.e., where each person could be involved with zero to many reports; and each report could involve one to many people). The associative class, therefore, in this case, “resolves” a many-to-many relationship. This, however, does not have to be the case. All that is necessary is that the associative class depends upon two or more classes for its existence. Thus, Person Identifier Assignment in Figure 5-8 (and more fully explained in the section on Role Patterns later in this document), is considered an associative class, because it depends on two other classes for its existence.
5.1.2.2.7 Event

A class is an event if and only if it models an “occurrent”; that is, something which “happens” in time.

Events are in contrast to the endurant stereotypes which model things that “exist” in time.

Additional characteristics of an “event” class:

- Events are expected to have a limited number of attributes. Typical event attributes are:
  - Name
  - Time (point in time, or beginning and ending time; specified as Date, Time, or DateTime)
  - Partitioning attributes for mutually exclusive subsets of events.

- Events never participate in dependency associations.

- Events never participate in realization inheritance. (There are no event categories.)

- Events may be specialized or generalized.

- Events may participate in whole/part relationships with other events.

- Events frequently are associated with associative classes via non-dependency associations. Associative classes typically represent relationships between classes, brought about and/or terminated by events.

Figure 5-8 An Example of an Associative Class That Does Not Resolve a Many-To-Many Relationship
• Events are also commonly associated with classes representing participants, locations, or endurants “affected by” the event.

How to model an “event” class:

• The stereotype <<event>> will be used to mark an “event” class.

• If the event class is associated with an associative class, the non-dependency association should be drawn from the associative class toward the event class.

• If the event class is associated with participants, locations or endurants “affected by” the event, the non-dependency associations should be drawn toward the event class.

PD Guidance 40
Select the [event] (event) icon from the Base Extension toolbar of PD and click in the diagram in which the class should appear.

Figure 5-9 presents examples of an event. The <<event>> class Employment represents a period of time during which an employer/employee relationship exists. This relationship itself is represented by the <<associative>> Employment Contract. The event has indirect relationships, via the associative, to two participant classes, Employer and Employee, and to a Location class. Were the <<associative>> class of no interest to the business, the relationships to the peripheral classes could be shown via non-dependency associations tied directly to Employment.
5.1.2.3 Class Stereotype Guidelines

Figure 5-10 depicts a “decision tree” to determine the stereotype of a class. This decision tree along with the set of questions in Figure 5-11 should be used to determine the stereotype of a class. In the figure, darker (orange) colored boxes (the leaves of the tree) are the stereotypes.
Does the class represent an enduring thing, or object, or does it represent an occurrence, or happening?

*If an occurrence, it is an <<event>> class.*
*If an enduring thing, it is a continuant.*

Does the continuant depend for its existence on zero, one, or two-or-more other classes?

*If one, it is a <<dependent>> class.*
*If two or more, it is an <<associative>> class.*
*If zero, it is an independent class.*

Is there a unified principle of identity for the independent class across all of its instances?

*If yes, it is a concrete class.*

Does the concrete class represent a permanent or transient concept?

*If permanent, it is a <<kind>> class.*
*If transient, it is a <<role>> class.*

*If no, it is an abstract class.*

Does the abstract class represent a permanent or transient concept?

*If permanent, it is a <<category>> class.*
*If transient, it is a <<role category>> class.*

**Figure 5-11 Set of Questions to Determine the Correct Stereotype**

Specifying the stereotype for a class provides the following benefits, some of which can also act as validation rules for the data model.

- Knowing a class’s stereotype communicates certain semantics that are true about the class (e.g., an independent class can “live” alone).

- Knowing a class’s stereotype can prevent a data modeler from drawing relationships that violate relationship rules that are based on the type. For example, the following rules must be adhered to:
A “role” class must specialize from a “kind” class or another “role” class

A “category” class must have at least two subclass classes that are either “kind” or “category” classes.

A “role category” class must have at least two subclass classes that are either “role” or “role category” classes.

Knowing a class’s stereotype can help a data modeler in assigning a name to the class. APPENDIX C (ELDM Naming Guidelines) contains naming guidelines that apply to specific types of classes.

5.1.3 Class Properties Based on Relationships to other Concepts

The Class concept also has the following properties, which are based on relationships to other concepts in the ELDM (as depicted above in Figure 5-1).

**Class Property: Consists of attributes (optional)**

The class’s “consists of attributes” property is a list of the attributes that are within the class. A class must have at least one attribute or one relationship in order to be fully defined. (Note that in Figure 5-1, this relationship is depicted indirectly from Class to Attribute via Attribute Construct.)

**Class Property: Owned by business area (mandatory)**

The class’s “owned by business area” property is the business area that “owns” the class (i.e., the business area within which the class has its primary purpose, as explained above within the section on Business Areas). For example, the Party and Party Name classes are owned by the Party business area. Although these classes may also exist on diagrams that are owned by other business areas, these classes primarily exist to support the Party business area.

**Class Property: Appears on diagram(s) (mandatory)**

The class’s “appears on diagram(s)” property is a list of all of the diagrams on which the class appears. A class must appear on at least one diagram, but may appear on many.
Note: The data modeling tool being used will most likely automatically keep track of this mapping, so the data modeler should not have to manually keep track of this information.

PD Guidance 43
The “appears on diagram(s)” property of a class is the set of diagrams found in the Class Properties Worksheet, Dependencies Tab, Diagram sub-tab, Name column.

Note: There may be multiple diagrams with the same name in the Model; the Parent column identifies the package [or model] within which this diagram appears.

Class Property: Contains association end(s) (optional)

The class’s “contains association end(s)” property is a list of the association ends with which the class is involved (i.e., each of which are part of an association relationship).

PD Guidance 44
The associations with which a class is involved are found in the Class Properties worksheet, Dependencies tab, Associations sub-tab, Name column. For each entry, to determine whether the class is involved with the “A” end, the “B” end, or both, look in the Class A or Class B column of that entry.

Class Property: Is superclass for inheritance relationship(s) (optional)

The class’s “is superclass for inheritance relationship(s)” property is a list of the inheritance relationships for which this class is a superclass.

Note: This list may include both specialization and realization relationships for which this class is a superclass.

PD Guidance 45
The “is superclass for inheritance relationship(s) property” of a class is the list of inheritance relationships found in the Class Properties worksheet, Dependencies tab, Generalizations sub-tab, Name column.

Note: The Generalizations sub-tab lists both kinds of inheritance relationships (i.e., both Specializations and Realizations) for which the class serves as a superclass [parent].

Class Property: Is subclass for inheritance relationship(s) (optional)

The class's “is subclass for inheritance relationship(s)” property is a list of the inheritance relationships for which this class is a subclass.

Note: This list may include both specialization and realization relationships for which this class is a subclass.
PD Guidance 46

The “is subclass for inheritance relationship(s) property” of a class is the list of inheritance relationships found in the Class Properties worksheet, Dependencies tab, Specializations sub-tab, Name column.

**Note:** The Specializations sub-tab lists both kinds of inheritance relationships (i.e., both Specializations [at most one] and Realizations [unbounded]) for which the class serves as a subclass [child]).

*Class Property: Uses metadata pattern(s) (optional)*

The class’s “uses metadata pattern(s)” property is a list of the Metadata Business Rules that apply to the class. APPENDIX G (Additional ELDM Guidelines) describes a “Broad Application of Metadata” pattern, which explains the “Metadata Pattern” concept and how it can be used by a class. (Note that this concept is not depicted in Figure 5-1, but rather in Figure G-4.)

PD Guidance 47

Metadata patterns are implemented using PD Business Rules.

The list of metadata patterns that the class uses can be found in the Class Properties worksheet, Rules tab.

See “Broad Application of Metadata” Pattern on page G-4.
6 Attribute Concept

An attribute is a characteristic, quality, or feature of the class. For example, the class **Person** might have the attributes: **name**, **weight**, and **height**.

Before an attribute is added to a class, the “candidate” attribute should first be modeled as a separate and related class (or at least the modeler should think about it this way, regardless of whether it is actually recorded this way in the data model). For example, if a data modeler had a **Person** class and wanted to model the skin color of a person, the data modeler should first model “skin color” in a separate and related class. Technically, modeling **Skin Color** as a separate class related to **Person** and modeling it as an attribute of **Person** are equivalent. Figure 6-1 shows “skin color” modeled as a separate and related class:

![Figure 6-1 Modeling an Attribute as a Separate and Related Class Initially](image)

The “candidate” attribute “skin color”, modeled as a separate class, should be inspected or tested to determine whether it should be brought in as an attribute or left as a separate class, as follows:

1. Is the concept a term that consists of two or more independent words? If so, each independent word should be considered as a candidate class. For example, “skin color” is a compound term. The data modeler should consider whether each concept should be modeled separately. For example, is the business interested in “skin”, separately from its “color”? If so, “skin” might need to be a class.

2. How important or significant is the concept to the business? For example, if the business has a great deal of interest in a person’s skin, the business may have multiple facts it wants to capture about skin, such as the skin’s texture type, hydration type, etc., and/or there may be a special relationship(s) between skin and another concept(s) in the data model. If the number of facts about the concept plus the number of unique relationships to the concept is two or more, the candidate attribute should remain as a separate class. Thus, **Person** might have a relationship to **Skin**, which has the attributes **color**, **texture type**, and **hydration type**.

3. Does the name of the relationship between the two concepts indicate that the two concepts are semantically the same concept or two different concepts? For example, the relationship in Figure 6-1 above is read as “Each Person has one and only one skin color”. The verb “has” does not provide much insight into whether skin color is a separate concept, since “has” can
mean “belongs to” among other things. The verb does not strongly imply that the two ends of the relationship should be separate concepts. Contrast this to the relationship “Each Person speaks one to many Languages”. In this case, even without the cardinality indicating that a person can have many languages, the semantics of the name of the relationship indicates that there are likely two distinct concepts involved: Person and Language, because the relationship name “speaks” indicates that Language is a different concept from Person.

4. What is the cardinality between the two concepts? If, after inspecting the concept as described above, the “maximum instances” property on the association’s cardinality in both directions is one (i.e., in Figure 6-1 above where each Person can have a maximum number of one instance of Skin Color; and where each instance of Skin Color can be for a maximum number of one Person), then this test would indicate that the candidate attribute should be brought in as an attribute (i.e., skin color should be brought into the class) as shown in Figure 6-2:

![Person class diagram]

**Figure 6-2 Bringing the Attribute into the Class**

If, on the other hand, the maximum instances property in either direction is many (or *), then this test would indicate the candidate attribute should remain as a separate class.

5. Is the attribute mandatory? Keep in mind the principle of “perfect knowledge” (as described in APPENDIX G: Additional ELDM Guidelines). Thus, if with perfect knowledge, the attribute is not mandatory in the class being examined, but it would be mandatory if placed in a subclass class, the candidate attribute should be placed in the subclass.

6. Does the candidate attribute represent a role of another class? For example, if the data modeler is considering adding an attribute to Device called device manufacturer (i.e., a role that an organization may play, indicating that the organization built a piece of equipment), a relationship should be drawn to a <<role>> class of the other class, as shown in Figure 6-3.
7. Would an instance of the class require more than one value for the candidate attribute? Keep in mind the principle of “snapshot in time” (as described in APPENDIX G: Additional ELDM Guidelines). For example, would an instance of Person require more than one value for skin color at a single point in time? If so, according to the first rule of normalization, the candidate attribute should be placed in a separate class.

Table 6-1 summarizes the above seven tests.

Table 6-1: Tests to Determine Whether Candidate Attribute Should be an Attribute or a Separate Class

<table>
<thead>
<tr>
<th>Test</th>
<th>Attribute of the Class</th>
<th>Separate Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compound Words</td>
<td>The business is not interested in each word/concept individually.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The business is interested in each word/concept individually.</td>
</tr>
<tr>
<td>2</td>
<td>Significance to the Business</td>
<td>Zero or one facts about or relationships to the candidate attribute</td>
</tr>
<tr>
<td>3</td>
<td>Relationship Name</td>
<td>Semantics of relationship name (generic, e.g. “has”) may indicate the candidate attribute is an attribute of the Class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Semantics of relationship name (specific, e.g. “manufactures”, “provides”) may indicate the candidate attribute should be a separate Class.</td>
</tr>
<tr>
<td>4</td>
<td>Cardinality on the Relationship</td>
<td>Maximum instances in both directions is one.</td>
</tr>
<tr>
<td>5</td>
<td>Mandatory Attribute</td>
<td>Attribute is mandatory.</td>
</tr>
<tr>
<td>6</td>
<td>Role of Another Class</td>
<td>Attribute does not represent a role of another class.</td>
</tr>
<tr>
<td>7</td>
<td>First Normal Form</td>
<td>There can only be one value for the candidate attribute for an instance of the class.</td>
</tr>
</tbody>
</table>

In addition, a candidate attribute should be inspected to ensure it is “atomic”. An atomic attribute represents a single fact that:
• Is meaningful to the business

• Contains no other additional fact(s) within it that are meaningful to the business that need to be parsed (or broken out into “sub-facts”).

For example, **height** and **quantity** are clearly both atomic attributes, since each one represents a single fact that is not parsed. Non-atomic attributes, however, often hide important business rules and facts and cause access to information to be lost or at least hindered. There are three common situations where non-atomic attributes are found and need to be resolved:

• **Multi-part attributes** - If the business cares about an attribute that contains multiple parts (e.g., a phone number, which consists of a country code, area code, exchange, etc.) and if the business recognizes one or more of the parts as a separate fact(s), each separate part that is recognized must be broken out as a separate attribute. For example, if the business cares about phone numbers and also recognizes the area code as a separate fact, then **area code** would need to be a separate attribute. It would still be appropriate to have a **phone number** attribute in addition to **area code**, if the full phone number was also treated as a fact by the business. Therefore, **phone number**, in this example, would be considered “atomic”, since there is no need to parse **phone number** in order to obtain the **area code**.

• **Codes** - The values for some attributes, codes in particular, might appear to represent one fact, however, often represent two or more facts. For example, an **asset type code** attribute may have “A” as a valid value, but this value indicates that the asset is a vehicle that has been leased from a commercial source for more than one year. This single attribute represents four distinct facts: the type of asset, whether it has been leased; the source of the lease; and the length of the lease. Each fact should be “broken out” as a separate attribute (or relationship to another concept) in order to comply with the atomic rule.

• **Comments** - Quite often, comments or text-block attributes contain business facts that should be defined as separate attributes (e.g., a comment field in a legacy system that contains “author name”). It may still be legitimate to have comments or text-block attributes as long as the attribute records a single business fact (e.g., **customer comments**).

Lastly, in general, derived attributes should not be included in an ELDM. A derived attribute is one for which the value could be calculated using the value(s) of other attribute(s). For example, if the “radius” of a circle were an attribute, there would be no reason to also include “diameter” as an attribute since its value could be calculated (i.e., by multiplying the “radius” by two). On the other hand, a modeler may decide to include as an attribute an “account balance” even though it is technically derivable, if the calculation to obtain its value is very involved.

```
For example, **height** and **quantity** are clearly both atomic attributes, since each one represents a single fact that is not parsed. Non-atomic attributes, however, often hide important business rules and facts and cause access to information to be lost or at least hindered. There are three common situations where non-atomic attributes are found and need to be resolved:

• **Multi-part attributes** - If the business cares about an attribute that contains multiple parts (e.g., a phone number, which consists of a country code, area code, exchange, etc.) and if the business recognizes one or more of the parts as a separate fact(s), each separate part that is recognized must be broken out as a separate attribute. For example, if the business cares about phone numbers and also recognizes the area code as a separate fact, then **area code** would need to be a separate attribute. It would still be appropriate to have a **phone number** attribute in addition to **area code**, if the full phone number was also treated as a fact by the business. Therefore, **phone number**, in this example, would be considered “atomic”, since there is no need to parse **phone number** in order to obtain the **area code**.

• **Codes** - The values for some attributes, codes in particular, might appear to represent one fact, however, often represent two or more facts. For example, an **asset type code** attribute may have “A” as a valid value, but this value indicates that the asset is a vehicle that has been leased from a commercial source for more than one year. This single attribute represents four distinct facts: the type of asset, whether it has been leased; the source of the lease; and the length of the lease. Each fact should be “broken out” as a separate attribute (or relationship to another concept) in order to comply with the atomic rule.

• **Comments** - Quite often, comments or text-block attributes contain business facts that should be defined as separate attributes (e.g., a comment field in a legacy system that contains “author name”). It may still be legitimate to have comments or text-block attributes as long as the attribute records a single business fact (e.g., **customer comments**).

Lastly, in general, derived attributes should not be included in an ELDM. A derived attribute is one for which the value could be calculated using the value(s) of other attribute(s). For example, if the “radius” of a circle were an attribute, there would be no reason to also include “diameter” as an attribute since its value could be calculated (i.e., by multiplying the “radius” by two). On the other hand, a modeler may decide to include as an attribute an “account balance” even though it is technically derivable, if the calculation to obtain its value is very involved.
```
6.1 Attribute Properties

Most of the Attribute Properties are found in one of the tabs of the Attribute Properties Worksheet, which can be viewed by double clicking the attribute in the browser. The appropriate tabs and fields are described later in this section or in APPENDIX B for the “Common Properties”.

Normally, you should view the Attribute Properties Worksheet with your “Favorite tabs” available; in other words, when viewing the Attribute Properties Worksheet, the word “More” should appear in the box in the lower left corner of the worksheet. If “Less” is displayed, click on the box to toggle the setting.

For ELDM work, the following tabs are the recommended favorites for the Attribute Properties Worksheet: General, Detail, Standard Checks, Also Known As, Rules, Requirements, Dependencies, and Version Info. To customize your favorite tabs, click on the Property Sheet Menu button to the right of the More {Less} box and select “Customize Favorite Tabs”. Ensure that each of the recommended favorites is checked. Unless you regularly work with PDMs or Requirement Models, you should uncheck the other options.

Note that the favorite tabs customization does not work as desired for all concepts. Some tabs, such as the Name Encoding and Physical Encoding tabs cannot be effectively removed from your favorites for certain worksheets. They are not needed for ELDM work, but will appear nevertheless. Also, some tabs that are required for ELDM work cannot be effectively added to your favorites for certain worksheets. In those cases, you will have to select More for that worksheet.

Note also that this customization applies to the way models are viewed at the workstation you are currently using. It does not change the model itself.

Figure 6-4, taken from Appendix A (ELDM Tool-Independent Metamodel), depicts the Attribute concept and its properties and relationships to other concepts.
6.1.1 Common Attribute Properties

The Attribute concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). Two of the common properties for an Attribute, the “name” and “definition” require the following additional explanation.

**Attribute Property: Name**

The name must be unique among attributes of the class. In other words, if the class Requirement has an attribute named description, there can be no other attributes within Requirement named “description”. This restriction carries over to subclasses of the class, which may not have an attribute with the same name unless that attribute is linked, via specialization, to the attribute in the parent class. However, there can be other classes, such as Report, not in the inheritance hierarchy, that have an attribute named “description”.

The attribute name must comply with ELDM Naming Guidelines (APPENDIX C) and the ELDM Acronym Guidelines (APPENDIX D).

In general, the names of all attributes for a class should be displayed (i.e., made visible) on diagrams. However, for some audiences, it may be appropriate to only show the class name (i.e., without displaying attribute names).

---

**Attribute Property: Definition (mandatory)**

The attribute definition must conform to APPENDIX E (ELDM Definition Guidelines).
6.1.2 Additional Attribute Properties

The Attribute concept has the following additional properties:

**Attribute Property: is mandatory (mandatory; “yes” or “no”)**

The attribute’s “is mandatory” property is a value to indicate whether the attribute would be populated if the business had perfect knowledge about the class, to include perfect knowledge about the attribute (see Appendix G: Additional ELDM Guidelines, “Perfect Knowledge” Pattern. However, if the attribute is still “optional” (i.e., even after applying the “perfect knowledge” principle), if appropriate, the data modeler should create a subclass of the class and move the attribute to the subclass, such that the attribute would be “mandatory”. For example, if the attribute **interest rate** is optional in the **Financial Account** class but is mandatory for a savings account, a **Savings Account** subclass class should be created and the attribute should be moved to **Savings Account** where it would be mandatory. This movement of the attribute to a subclass in order to make it mandatory should especially be done if:

- The subclass is important to the business (i.e., this more detailed model communicates an important business rule)

- It is practical to create the subclass (i.e., without overly complicating the data model).

However, if the attribute is still not “mandatory” (i.e., despite having perfect knowledge and regardless of whether it was appropriate to move the attribute to a subclass), this property should be set to “optional”. For example, if a class has an **end date** attribute, if with perfect knowledge the “end date” might not be populated, and if it is not appropriate to subclass the class, the **end date** attribute should be marked as “optional”.

**Attribute Property: is derivable (derivable; “yes” or “no”)**

The attribute’s “is deliverable” property is a value to indicate whether the value of the attribute could be derived from the value(s) of other attribute(s) in the model. As stated above, such attributes are generally not included in an ELDM, but if included they should be designated as such.
the attribute is derivable, the algorithm for deriving the value should be specified in the Algorithm text box of the Attributes Properties worksheet, Derivation tab.

6.1.3 Attribute Properties Based on Relationships to other Concepts

The Attribute concept also has the following properties, which are based on relationships to other concepts in the ELDM:

**Attribute Property: Partition for mutually exclusive inheritance relationships (optional)**

A brief explanation of the “inheritance set” and “mutually exclusive inheritance relationships” concepts follows. For a more detailed explanation, refer to section 7.1.2.1.1.3 below. A partitioning attribute is an attribute that partitions an inheritance set, that is, that allows an instance of a superclass class to be designated as being an instance of one and only one of its subclasses that is part of the inheritance set. The partitioning attribute’s “partition for mutually exclusive inheritance relationships” property is a list of the mutually exclusive specialization and realization relationships which belong to the inheritance set for which this attribute acts as a partitioning attribute. A separate partitioning attribute is needed for each independent group of subclasses (i.e., for each “partition” of the superclass).

PD Guidance 54

The “partition for mutually exclusive inheritance relationship” property of a partitioning attribute is the list of names in that attribute’s Attributes Properties worksheet, Dependencies tab, Specified Generalizations sub-tab, Name column. (The Specified Generalizations sub-tab – which would be more appropriately called the Specified Inheritances sub-tab – will have the list of realizations and specializations belonging to the Inheritance Set for which the attribute is a partition.)

**Attribute Property: Described by data type (mandatory)**

The attribute’s “described by data type” property is the format or manner in which an attribute’s values should be represented in order to be meaningful to the user. For example, the attribute **begin date** would be expected to have the primitive data type “date”; and an order line item number would be expected to have the domain data type “positive integer”.

PD Guidance 55

The “Described by data type” property can be set from the Attributes Properties worksheet, General tab, Data Type field.

The specific way the data type property is set depends on which of the five data type concepts, described in section 8, is used. Those concepts are Primitive, Domain, Enumerated List, Data Structure, and Union.

If the data type is to be one of the eight Primitive types, it can be chosen from the pull-down list in the Data Type field.
If the data type is to be a Domain, Enumerated List, Data Structure, or Union, it can be chosen by opening the Select data type window by clicking the Datatypes button to the right of the Data Type field. You may then choose either the Browse or Index tab to locate the specific class which defines the Domain, Enumerated List, Data Structure, or Union desired.

**Attribute Property: Part of data structure (optional)**

The attribute’s “part of data structure” property is the data structure for which this attribute acts as one of its mandatory attributes. Attributes almost always reside in, or are part of, a class. The only exceptions are those attributes that are part of a “data structure” (see explanation of a data structures in section 8.2.4). Thus, each attribute must be part of either a class or a data structure.

PD Guidance 56

If an attribute acts as a mandatory attribute of a data structure, its “Part of data structure property” can be found in the Attributes Properties worksheet, General tab, Parent field.

**Attribute Property: Part of class (optional)**

The attribute’s “part of class” property is the class in which the attribute exists. Attributes almost always reside in, or are part of, a class. The only exception is those attributes that are part of a “data structure” (see explanation of a data structures in section 8.2.4). Thus, each attribute must be part of either a class or a data structure.

PD Guidance 57

If an attribute exists within a class, its “Part of class” property can be found in the Attributes Properties worksheet, General tab, Parent field.

**Attribute Property: Uses metadata pattern(s) (optional)**

The attribute’s “uses metadata pattern(s)” property is a list of the metadata patterns that the attribute uses. APPENDIX G (Additional ELDM Guidelines) describes a “Broad Application of Metadata” pattern, which explains the “Metadata Pattern” concept and how it can be used by an attribute.

PD Guidance 58

Metadata patterns are implemented using PD Business Rules.

The list of metadata patterns that the attribute uses can be found in the Attribute Properties worksheet, Rules tab, Name column.

See “Broad Application of Metadata” Pattern” on page G-4.
Attribute Property: Has attribute alias (optional)

The attribute’s “has attribute alias” property is a list of alternate (or alias) names that the business uses for the attribute along with the context or portion of the business where that name is used.

PD Guidance 59

The Attributes Properties worksheet, Also Known As tab, aka field can be used to enter free formatted text listing attribute aliases along with business usage or other context.

Attribute Property: Specializes (optional)

The attribute’s “specializes” property shows the parent attribute that serves as a more abstract form of this attribute. For example, as shown in Figure 6-5, the class IMF Financial Account has an IMF account number attribute that has a domain that restricts it to be 20 characters in length. This attribute specializes the Financial Account account number attribute, which has a domain that allows a string up to 40 characters in length. The more abstract form of the account number attribute may be used for certain parts of the enterprise that do not distinguish between the kinds of financial accounts.

Typically each part of the enterprise will either use the more abstract form or the more detailed form of the attribute, but not both.

PD Guidance 60

To set an attribute’s “specializes” property, use the Attribute worksheet, General tab, Other section, and select the appropriate (more abstract) attribute from the pull-down list of the Specializes field.

![Diagram of category and kind]

Figure 6-5 Example of Specialized and Generalized Attributes

Attribute Property: Specialized as (optional)

The attribute’s “specialized as attribute(s)” property is a list of attributes for which this attribute is a more abstract (or more generalized) representation. For example, as shown in Figure 6-5, the class
Financial Account may have an account number attribute with a domain that allows a value up to 40 characters in length to handle the longest account number of any of its children classes. This attribute could have a specialized attribute in the IMF Financial Account class called IMF account number that has a domain that limits its length to 20 characters. The specialized form of the IMF account number attribute may be used by certain parts of the enterprise that require the more detailed form.

Typically each part of the enterprise will either use the more abstract form or the more detailed form of the attribute, but not both.

- PD Guidance 61
  - The “specialized as attribute(s)” property of an attribute can be found in the Attribute Properties worksheet, Dependencies tab, Extended Inverse Collections Sub-tab. The list of attributes for which this attribute is a more abstract representation is the set of attributes in the Collection Owner column which have a corresponding entry of “specialization” in the Inverse Collection Name column.
7  Relationship Concept

A relationship is a mechanism to communicate a business need to directly associate the information in one class with the information in another class. A relationship allows the “business” (i.e., anyone in the business), when it knows about one class, to be able to directly find out related information in another class. For example, if the business needs to capture information about people (e.g., in the Person class) and needs to capture or retrieve information about the skills of people (e.g., in the Person Skill class), then a relationship is needed between Person and Person Skill.

A relationship usually involves two (and only two) classes. The only exception to this rule is for a recursive relationship, which involves only one class. A “recursive” relationship is a relationship from and to the same class. For example, as shown in Figure 7-1, a recursive relationship named “spawns” on the class Process would indicate that one process can spawn other processes.

![Figure 7-1 Example of a Recursive Relationship](image)

All relationships should be shown using “rectilinear” lines (i.e., lines that are vertically or horizontally straight that may also have one or more 90 degree angles).

There are two kinds of relationships: associations, and inheritances. In turn, associations have one of four stereotypes, dependency, non-dependency, composition, and aggregation; and inheritances have one of two stereotypes, specialization or realization. The kinds of relationships and the various stereotypes are discussed in detail in the following sections.

7.1  Relationship Properties

Most of the Relationship Properties are found in one of the tabs of the appropriate relationship Properties Worksheet (Association, Specialization or Realization), which can be viewed by double clicking that relationship in the browser. The appropriate tabs and fields are described later in this section or in APPENDIX B for the “Common Properties”.

Normally, you should view the appropriate relationship Properties Worksheet with your “Favorite tabs” available; in other words, when viewing the relationship Properties Worksheet, the word “More” should appear in the box in the lower left corner of the worksheet. If “Less” is displayed, click on the box to toggle the setting.
For ELDM work, the following tabs are the recommended favorites for the Association Properties Worksheet: General, Detail, Also Known As, Requirements, Dependencies, and Version Info, while the following tabs are recommended favorites for both Realization and Specialization Worksheets: General, Also Known As, Requirements, Dependencies, and Version Info. To customize your favorite tabs, click on the Property Sheet Menu button to the right of the More {Less} box and select “Customize Favorite Tabs”. Ensure that each of the recommended favorites is checked. Unless you regularly work with PDMs or Requirement Models, you should uncheck the other options.

Note that the favorite tabs customization does not work as desired for all concepts. Some tabs, such as the Name Encoding and Physical Encoding tabs cannot be effectively removed from your favorites for certain worksheets. They are not needed for ELDM work, but will appear nevertheless. Also, some tabs that are required for ELDM work cannot be effectively added to your favorites for certain worksheets. In those cases, you will have to select More for that worksheet.

Note also that this customization applies to the way models are viewed at the workstation you are currently using. It does not change the model itself.

Figure 7-2 below, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Relationship concept and its properties and relationships to other concepts.

![Figure 7-2 The Relationship Concept and Its Properties and Relationships](image-url)
7.1.1 Common Relationship Properties

The Relationship concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). Some of the common properties require additional explanation, as they pertain to specific Relationship kinds or stereotypes. Those additional explanations can be found in the appropriate sections below pertaining to those kinds or stereotypes.

7.1.2 Additional Relationship Properties

**Relationship Property: Type (mandatory)**

A relationship’s “type” property identifies the kind of the relationship, either association or inheritance.

---

**PD Guidance 63**

Associations are automatically assigned the type “association” by PD. Realizations and specialization relationships are automatically assigned the type “inheritance,” although the inheritance type is not visible to the user.

---

**Relationship Property: Stereotype (mandatory)**

A stereotype is a property of the relationship used to indicate that the relationship conforms to a certain set of characteristics. For an association, the stereotype values are either “dependency”, “non-dependency”, “composition”, or “aggregation” and for an inheritance, the stereotype values are either “specialization” or “realization”, as depicted in Table 7-1:

<table>
<thead>
<tr>
<th>Type</th>
<th>Stereotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inheritance</td>
<td>1. Specialization 2. Realization</td>
</tr>
<tr>
<td>Association</td>
<td>3. Dependency 4. Non-Dependency</td>
</tr>
<tr>
<td></td>
<td>5. Composition 6. Aggregation</td>
</tr>
</tbody>
</table>

---

**PD Guidance 64**

The relationship “Stereotype” property is found in the General tab, Stereotype field of the Association, Specialization, or Realization worksheet, as appropriate, and can be modified using the pull-down list.

7.1.2.1 Relationship Details

The Inheritance Relationship Type and its two stereotypes and various properties, are described in detail in section 7.1.2.1.1. The Association Relationship Type and its four stereotypes and various properties, are described in detail in section 7.1.2.1.2.
7.1.2.1.1 Inheritance Relationship Type

An inheritance is a superclass-subclass relationship between two distinct classes. In other words, it is a non-recursive relationship in which all instances of one class (the child, or subclass) are instances of the other class (the parent, or superclass).

- As a result, all of the attributes and relationships of the superclass of an inheritance relationship apply to (or are “inherited” by) the subclass.

7.1.2.1.1.1 Realization Relationship

A realization relationship is an inheritance relationship between two classes in which:

- The superclass is a category or role category (and therefore abstract) and the subclass is concrete.

We say that the child class is a realization of the parent via this relationship, and that the parent is a categorization of the child via this relationship.

Figure 7-3 shows two “realizations”: one between Party and Person, and one between Party and Organization.

![Diagram showing Party and Person realization](image)

**Figure 7-3 Sample Realizations**

PD Guidance 65

To draw a realization inheritance relationship from Child Class A to Parent Class B, select the (inheritance, but mislabeled as generalization) icon from the Palette toolbar of PD. (N.B. Do not select the standard PD realization icon for realization inheritance relationships. Use the icon.) Then click...
The following special rules apply to “realizations” only:

- Although not graphically shown on diagrams, the cardinality of realizations must be 1..1 and 0..1 (with the 1..1 being on the superclass side and 0..1 being on the subclass side).

- Although not named, all realizations could be interpreted as if named “acts as a”, where the subclass “acts as a” superclass (e.g., Organization acts as a Party). Thus, the name should be left blank if the data modeling tool will allow this; or may be named “acts as a”, if the tool requires a name. Regardless of whether the tool requires a name, the name should not be shown on diagrams. For example, as shown in Figure 7-3, although no name is shown, the relationship on the right of the figure can be read as: each Organization “acts as a” Party.

- Classes are allowed to have multiple inheritance via realization, meaning that a given subclass can have multiple superclasses with which it participates via “realizations”. The subclass will inherit the properties from all of the superclasses. However, since the superclasses in realization relationships are <<categories>> or <<role categories>>, which do not themselves have an identity, there is no problem of the subclass having an ambiguous identity. For example, as shown in Figure 7-4, the Person <<kind>> (subclass) class has a realization relationship to the Party <<category>> (superclass) and a realization relationship to the Intelligent Agent <<category>> (superclass). The Person <<kind>> class has its own identity.

![Diagram showing multiple inheritance allowed via realizations]

*Figure 7-4 Multiple Inheritance Allowed Via Realizations*

Table 7-2 depicts valid realizations (i.e., the stereotypes of classes that may act as “subclass” and “superclass” classes using a realization relationship). This table shows the following:

- Only a kind may be the child in a realization relationship to a category.
• Only a role may be the child in a realization relationship to a role category.

### Table 7-2 Valid Realizations

<table>
<thead>
<tr>
<th>Superclass</th>
<th>Kind</th>
<th>Role</th>
<th>Category</th>
<th>Role Category</th>
<th>Dependent</th>
<th>Associative</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td></td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td></td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role Category</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 7.1.2.1.1.2 Specialization Relationship

A **specialization relationship** is an inheritance relationship in which:

- Both classes have the same stereotype or

- The supertype is a <<kind>> and the subtype is a <<role>> or

- The supertype is a <<category>> and the subtype is a <<role category>>.

We say that the child class in the inheritance is a **specialization class** of the parent class via this relationship, and that the parent is a **generalization class** of the child via this relationship.

**PD Guidance 66**

To draw a specialization inheritance relationship from Child Class A to Parent Class B, select the (inheritance, but mislabeled as generalization) icon from the *Palette* toolbar of PD. Then click on Class A and hold the mouse button down as you draw a line to Class B. An appropriate connector will appear when you release the mouse button.

The following special rules apply to “specialization relationships”:

- Although not graphically shown on diagrams, the cardinality of specialization relationships must be 1..1 on both ends.

- Classes are not allowed to have **multiple inheritance via specialization**, if any of its superclasses have different principles of identity. To have two or more superclass classes with different principles of identity would mean that the subclass would inherit the principle of identity from each such superclass. This is not allowed, since it would indicate confusion as to what the subclass was. For example, Figure 7-5 shows an illegal specialization in which the **Hybrid** <<class>> would need to inherit identity from the disparate identities of the **Person** <<class>> and the **Organization** <<class>>.
Classes are allowed to have multiple inheritance via specialization, as long as all superclasses have the same unified principle of identity. Such a circumstance provides no confusion as to the identity of the subclass. For example, Figure 7-6 shows a legal specialization in which the Male Native Born Citizen <<class>>, as well as all the other classes in the figure, inherit their principles of identity from the Person <<class>>.

Figure 7-6: Legal Multiple Inheritance via Specialization
• Note that a class can have inheritance via specialization as well as (possibly multiple) inheritance via realization.

• A “specialization relationship” can be created with any stereotype of classes acting as the superclass.

• The child class of a “specialization relationship” is always of the same stereotype as its parent, except that, in addition, a <<role>> can be a subclass of a <<kind>>, and a <<role category>> can be a subclass of a <<category>>.

**Note:** In all cases, the subclasses must comply with all of the rules established for subclass classes stated elsewhere in this document.

Table 7-3 depicts valid specialization relationships (i.e., the stereotypes of classes that may act as “subclass” and “superclass” classes using a specialization relationship).

### Table 7-3 Valid Specialization Relationships

<table>
<thead>
<tr>
<th>Superclass</th>
<th>Kind</th>
<th>Role</th>
<th>Category</th>
<th>Role Category</th>
<th>Dependent</th>
<th>Associative</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td></td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role Category</td>
<td></td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Valid</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Valid</td>
</tr>
</tbody>
</table>

### 7.1.2.1.1.3 Inheritance Sets and Independent Inheritance

There is an additional subdivision of inheritance relationships:

1. An inheritance relationship which is a member of an inheritance set (may be either specialization or realization) or

2. An independent inheritance relationship (only applies to specializations).

#### 7.1.2.1.3.1 Inheritance Sets

An inheritance set is a set of two or more inheritance relationships where each instance of the superclass (the parent class) may be an instance of at most one of the subclasses (the children classes) via that set of inheritance relationships. In other words, an instance of any one subclass (via that set of inheritance relationships) is excluded from also being an instance of any other one of the subclasses. The inheritance relationships which are members of the inheritance set are said to be *mutually exclusive.*
How to model an inheritance set (mutually exclusive inheritance relationships):

Assume the existence of a superclass and some number of subclasses linked to that superclass by a set of inheritance relationships. To explicitly designate the set of a superclass and some number of subclasses linked to that superclass by a set of inheritance relationships. To explicitly designate the set of inheritance relationships as an inheritance set, add an attribute to the superclass, and link to that attribute each of the inheritance relationships meant to be members of the inheritance set. This attribute will be referred to as the partitioning attribute of the inheritance set. This linkage should be made via a property of the inheritance relationship.

In addition, and for greater visibility of the total membership of the inheritance set, the modeler may build an enumerated list containing the names of the subclasses, and type the partitioning attribute to the enumerated list.

For clarity, the metamodel diagram depicted in Appendix A, and the examples shown throughout this document, display the enumerated lists linked to partitioning attributes.

**PD Guidance 67**
To establish an inheritance set, create an attribute of the superclass to be used as the partitioning attribute. For each inheritance relationship which is to belong to the inheritance set, do the following:

- Open the inheritance’s Properties worksheet (either a Specialization Properties worksheet or a Realization Properties worksheet). In the General tab, Other section, click on the down arrow for the Partitioning Attribute field and select the appropriate entry to serve as the partitioning attribute. Use the same attribute as the partitioning attribute for the each member of the set.

Note that partitioning attributes are displayed in *bold italics* font.

**Local Guidance 4**
Local guidance eliminates the need for the enumerated list within the UML logical data model and calls for the partitioning attribute to be typed “auto”. (All partitioning attributes should be datatyped “auto” and only partitioning attributes should be datatyped "auto".) When the Autogen program generates XML Schema from the UML model, it will build the enumerated list and properly type the partitioning attribute.

An inheritance set is said to be complete if and only if every instance of the superclass must be an instance of exactly one of the subclasses; that is, the modeler has fully indicated all the possible subclasses. An inheritance set may consist of all specializations, all realizations, or a mixture of specializations and realizations.

Figure 7-7 depicts an example of an inheritance set. This model asserts the following:

- Each instance of a financial account may be either a savings account or checking account, but cannot be both
- All savings accounts and all checking accounts are financial accounts
All of the attributes and relationships for Financial Account are inherited by both Savings Account and Checking Account.

![Figure 7-7 Sample “Inheritance Set”](image)

If the modeler asserts, via an additional constraint, that the inheritance set in Figure 7-7 is complete, then the first bullet above is replaced by the following:

- Each instance of a financial account must be either a savings account or checking account, but cannot be both.

Additional guidance regarding modeling of inheritance sets:

- All connectors in an inheritance set should be connected in a “pitchfork” configuration, by overlapping the arrowheads one atop the other, as shown in Figure 7-7.

- A separate partitioning attribute is needed for each inheritance set. A partitioning attribute is an attribute of the superclass class that allows an instance of the superclass to be designated as being an instance of one of its subclasses as defined by an inheritance set. Figure 7-7 shows the partitioning attribute “type” within the Financial Account superclass class.

- The values of the partitioning attribute must exactly match the names of the subclasses (to include having the same case, such as “Title Case”, as the subclass names). In Figure 7-7, the legal values of “type” are “Savings Account” and “Checking Account”.

- Each inheritance relationship within an inheritance set must identify the associated partitioning attribute. For example, in Figure 7-7, the specialization relationship between Financial Account and Savings Account and the specialization relationship between Financial Account and Checking Account must identify the attribute “type” as the associated partitioning attribute.
• One superclass class may have multiple inheritance sets. This is referred to as *multiple partitioning* or *orthogonal inheritance sets*. In this case, a separate partitioning attribute should be placed within the superclass for each partition, as shown in Figure 7-8, depicting three partitioning attributes and three inheritance sets.

![Diagram showing multiple partitioning and inheritance sets](image)

**Figure 7-8 Example of Multiple Partitioning**

• A given category or role category can have at most one inheritance set which contains realizations, whether that inheritance set contains only realizations or a mixture of realizations and specializations. However, the category or role category can have additional inheritance sets which contain only specializations.

The Inheritance Set concept has the following property, which is based on a relationship to another concept in the ELDM:

**Inheritance Set Property: Has partitioning attribute (mandatory)**

The inheritance set’s “has partitioning attribute” property is the attribute that acts as the partitioning attribute for the set of mutually exclusive specializations and realizations.

PD Guidance 68

The “has partitioning attribute” property of an inheritance set can be found in the Specialization (or Realization) Properties worksheet (of any of the inheritance relationships belonging to the set), General tab, Other section, Partitioning Attribute field. See PD Guidance 67 for additional information on setting this attribute.
**Inheritance Set Property: Is complete (optional)**

The inheritance set’s “is complete” property is a Boolean indicator that every member of the inheritance set is known and has been specified in the model. If this property is not specified as true, it may be assumed that the inheritance set is not complete.

**PD Guidance 69**

To set the inheritance set property to complete, open the partitioning attribute’s Attribute Properties worksheet. On the Rules tab, select the (Add Objects) icon. A Selection worksheet will open, containing a list of Business Rules associated with the model. Check the selection box next to the Complete business rule.

**Inheritance Set Property: Has inheritance relationships (mandatory)**

The inheritance set’s “has inheritance relationships” property is the set of mutually exclusive inheritance relationships (realizations and specializations) belonging to the set (i.e. associated to the set via the set’s partitioning attribute).

**PD Guidance 70**

The set of mutually exclusive inheritance relationships for an inheritance set can be found in the set’s partitioning attribute’s Attribute Properties worksheet, Partitioned Inheritances tab, Name column. The set is comprised of all specializations and realizations found in that column.

7.1.2.1.3.2 Independent Inheritance

An independent specialization relationship is a specialization relationship where each instance of the superclass could be an instance of the subclass regardless of whether it is also an instance of any other subclass. This scenario is common for <<role>> subclasses. Figure 7-9 depicts an example of this type of specialization. This model asserts the following:

- Each instance of a person could be a doctor and each instance could be a lawyer
- All doctors and all lawyers are persons
- All of the attributes and relationships for Person are inherited by both Doctor and Lawyer.
How to model a “specialization with independent subclass”:

- A partitioning attribute should not be included in the superclass.

- The relationship should not be named explicitly; the name is always “is a”, as in the subclass “is a” superclass. Thus, the name should be left blank if the data modeling tool will allow this, or may be named “is a”, if the tool requires a name. Regardless of whether the tool requires a name, the name should not be shown on diagrams. For example, as shown in Figure 7-9, although no name is shown, the relationship on the left side of the figure can be read as: “Each Doctor is a Person”.

7.1.2.1.1.4 Inheritance Relationship Properties

The Inheritance concept has the following properties which are based on relationships to other concepts in the ELDM:

Inheritance Property: Has superclass class (mandatory)

The inheritance’s “has superclass class” property is the class that acts as the superclass or parent of the inheritance.

PD Guidance 71

The “has superclass class” property of an inheritance can be found in the appropriate (Specialization Properties or Realization Properties) worksheet, General tab, Other section, Parent Object field.

Inheritance Property: Has subclass class (mandatory)

The inheritance’s “has subclass class” property is the class that acts as the subclass or child of the inheritance.
The “has subclass class” property of an inheritance can be found in the appropriate (Specialization Properties or Realization Properties) worksheet, General tab, Other section, Child Object field.

### 7.1.2.1.1.5 Special Guidelines and Rules for Inheritance Relationships

The guidelines for “naming” inheritance relationships (i.e., both specializations and realizations) are shown in Table 7-4.

#### Table 7-4 Summary of Naming Guidelines for Specializations and Realizations

<table>
<thead>
<tr>
<th>Inheritance Stereotype</th>
<th>Name the Inheritance</th>
<th>Show the Name</th>
<th>Name Value</th>
<th>Meaning of Inheritance</th>
<th>Arrow Head in Middle of Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specialization</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
<td>&quot;is a&quot;</td>
<td>&quot;acts as a&quot;</td>
</tr>
<tr>
<td>Realization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

The following special rules apply to both “specialization” and “realization” relationships:

- Each subclass class involved in a “specialization” or “realization” relationship must have at least one of the following two characteristics:
  1. A unique attribute or
  2. A unique relationship (i.e., “a relationship with another class that is logically correct only for it” and no other class)."

- **Nested inheritance** is allowed, meaning that a child (via either a “specialization” or “realization”) may also be a parent to another child (also via either a “specialization” or “realization”, as appropriate for its stereotype). A class that plays the role of both superclass and subclass must adhere to the guidelines in this document for both. A nested inheritance structure allows a data modeler to show precise business rules. For example, the nested inheritance structure in Figure 7-10 allows a data modeler to specify attributes and relationships that are:
  
  - Common for all financial accounts
  - Common for all checking accounts (but unique from other types of financial accounts)
  - Common for all interest-bearing checking accounts (but unique from other types of checking accounts)
  - Common for all non-interest-bearing checking accounts (but unique from other types of checking accounts).

**Note:** Two or more “levels” of inheritance is referred to as “nested inheritance.”
7.1.2.1.2 Association Relationship Type

An association is any relationship that is not an inheritance relationship.

There are four stereotypes of associations: dependency, non-dependency, composition, and aggregation. Those stereotypes are discussed in detail in sections 7.1.2.1.2.1, 7.1.2.1.2.2, 7.1.2.1.2.3.1, and 7.1.2.1.2.3.2 below.

Properties of the association concept (all four stereotypes) are discussed in section 7.1.2.1.2.4 below.

7.1.2.1.2.1 Dependency Association

A dependency association is an association where one class in the relationship depends for its existence on the other class.

Figure 7-11 depicts dependency associations drawn from Class A to Class B. Class B will be either a <<dependent>> class or an <<associative>> one.

Figure 7-10 Example of Nested Specialization
Figure 7-12 depicts an example of a dependency association where **Person Skill** is a <<dependent>> class. In this example **Person Skill** depends for its existence on **Person**.

![Figure 7-12 Dependency Association](image)

Additional characteristics that always apply to dependency associations:

- **Cascade delete** – For each dependency association, if an instance of the class, upon which the other class depends for its existence, were deleted, all of the corresponding instances of the dependent or associative class would also have to be deleted. Thus, in the example above, if an instance of **Person** were deleted, all of the corresponding instances of **Person Skill** would also have to be deleted.

- **“B” is non-sharable** – A single instance of the dependent/associative class cannot be shared by another instance of the class upon which it depends for its existence. Thus, a single instance of **Person Skill** in Figure 7-12 cannot be shared by another instance of **Person**.

How to model a dependency association:

- It should be depicted with a solid line (i.e., not dashed)

- It must have 1..1 cardinality on the “A” side (i.e., the side of the class assuring the existence of the other)

- It can have any of the four valid cardinalities on the “B” side (i.e., 1..*, 0..*, 1..1, 0..1) or in rare scenarios the cardinality may reflect specific numbers (as discussed previously)

- It should be named in the direction from the side where the cardinality is 1..1 to the other side (i.e., from “A” to “B”, as described above)

- Class “B” must be either a <<dependent>> or <<associative>> class.

---

**PD Guidance 73**

To draw a dependency association from Class A to Class B, select the **dependency** icon from the **Base Extensions** toolbar of PD. Then click on Class A and hold the mouse button down as you draw a line to Class B. An appropriate connector will appear when you release the mouse button.
Table 7-5 depicts valid dependency associations (i.e., the stereotypes of classes that may be related to other stereotyped classes using a dependency association). This table shows that a class of any endurant stereotype may provide existence to either a dependent or associative class via a dependency association.

<table>
<thead>
<tr>
<th>To From</th>
<th>Kind</th>
<th>Role</th>
<th>Category</th>
<th>Role Category</th>
<th>Dependent</th>
<th>Associative</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role Category</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependent</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Associative</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.2.1.2.2 Non-dependency Association

A non-dependency association is an association where neither class in the relationship depends for its existence on the other class. Thus, if either class were deleted, the corresponding instances of the other class would not have to be deleted.

How to model a non-dependency association:

- It should be depicted with a dashed line and
- It can have any of the four valid cardinalities on either side (i.e., 1..*, 0..*, 1..1, 0..1) or in rare scenarios the cardinality may reflect specific numbers (as discussed previously).

To draw a non-dependency association from Class A to Class B, select the (non-dependency) icon from the Base Extensions toolbar of PD. Then click on Class A and hold the mouse button down as you draw a line to Class B. An appropriate connector will appear when you release the mouse button.

Figure 7-13 depicts an example of a non-dependency association.
Figure 7-13 Example of a Non-Dependency Association

Table 7-6 depicts valid non-dependency associations (i.e., the stereotypes of classes that may be related to other stereotyped classes using a non-dependency association). This table shows that a class of any stereotype may be connected to a class of any stereotype via a non-dependency association.

Table 7-6 Valid Non-Dependency Associations

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Kind</th>
<th>Role</th>
<th>Category</th>
<th>Role Category</th>
<th>Dependent</th>
<th>Associative</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Kind</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>Role</td>
<td>Role</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>Category</td>
<td>Category</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>Role Category</td>
<td>Role Category</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>Dependent</td>
<td>Dependent</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>Associative</td>
<td>Associative</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
<tr>
<td>Event</td>
<td>Event</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
</tr>
</tbody>
</table>

7.1.2.1.2.3 Whole/Part Associations

A **whole** class is a class an instance of which is an entirety consisting of pieces or members.

A **part** class is a class an instance of which is a piece, segment, or member of something which combined with others makes up a whole.

A **whole/part association** is an association between a whole and a part representing the claim that any instance of the whole consists of instances of the part.

A whole/part association is said to be **shareable** if and only if a single instance of the part class can simultaneously belong to two or more instances of the whole class (via this association). Conversely, a whole/part association is said to be **non-shareable** if and only if a single instance of the part class cannot simultaneously belong to two or more instances of the whole class (via this association).

For example, an automobile engine can belong to only one automobile at a time, so the association between the **Automobile** whole class and the **Automobile Engine** part class is non-shareable. On the other hand a person can belong to multiple organizations simultaneously, so the association between **Organization** whole class and **Person** part class is sharable.
The stereotype **aggregation** is applied to a shareable whole/part association. The stereotype **composition** is applied to a non-shareable whole/part association. Instructions for drawing these stereotypes, as well as additional stereotype details, are provided in sections 7.1.2.1.2.3.1 and 7.1.2.1.2.3.2 below.

Note that the concept of shareability refers to a single association. In different words, this concept of shareability can be referred to as **local exclusion**. Non-shareability does not preclude a part class from having whole/part associations with two different whole classes. In this case, even if both associations were non-shareable, an instance of the part class could simultaneously belong to one instance of each whole class. For example, a heart is both part of a body, and non shareable among bodies, and a part of a circulatory system, but non shareable among circulatory systems. (This methodology does not address the concept of global exclusions.)

A whole/part association is said to be **inseparable** if and only if the whole is essential to the part; i.e., an instance of the part class cannot be removed from the instance of the whole class without the part ceasing to exist.

For example, we might have a simple model of the Earth as being composed of a core and its surrounding oceans and continents. We cannot remove an ocean or continent from the Earth (never mind the physical difficulty) without the ocean or continent ceasing to be an ocean or continent. The association between the whole class and the part class in this case is inseparable.

A whole/part association is said to be **essential** if and only if the part is essential to the whole; i.e., an instance of the part class cannot be removed from the instance of the whole class without the whole ceasing to exist.

For example, a water molecule is composed of two hydrogen atoms and an oxygen atom. None of the atoms can be removed from the water molecule without that water molecule ceasing to exist. The association between the class **Water Molecule** and the class **Oxygen Atom** is essential.

As a special case, when the whole class is a role, we use the **immutable**, rather than essential, for a whole/part association when the part is essential to the whole.

For example, although a person can exist without arms, a person cannot participate in standard boxing without arms. Therefore, although the association from the class **Body** to the class **Arm** is not essential, the association from the class **Boxer Body** to the class **Arm** is immutable.

The following six samples reiterate the above concepts by combining various whole/part properties and cardinalities. All six samples show non-shareable (composition) whole/part associations. Therefore, in each case, an instance of the B class can belong to at most one instance of the A class. For these explanations, cardinality is indicated for only one end (whole or part) of the association.

In Figure 7-14, the whole is optional. An instance of class B can exist without any instance of class A being associated with it.
In Figure 7-15, the whole is mandatory. An instance of class B can exist only if it belongs to some instance of class A.

In Figure 7-16, the whole is mandatory and the part is inseparable. An instance of class B can exist only if it belongs to a particular fixed instance of class A.

In Figure 7-17, the part is optional. An instance of class A can exist without any instance of class B being associated with it.

In Figure 7-18, the part is mandatory. An instance of class A can exist only if it contains at least one instance of class B.

In Figure 7-19, the part is mandatory and the part is essential. An instance of class A can exist only if it contains one (or more) fixed instance(s) of class B.
Figure 7-19 Composition with Mandatory and Essential Part

Table 7-7 depicts valid whole/part associations (i.e., the stereotypes of classes that may be related to other stereotyped classes using a whole/part association). It shows that only independent classes can participate in whole/part associations with other independent classes, and that events can only participate in whole/part associations with other events.

Table 7-7 Valid Whole/Part Associations

<table>
<thead>
<tr>
<th>Whole</th>
<th>Kind</th>
<th>Role</th>
<th>Category</th>
<th>Role Category</th>
<th>Dependent</th>
<th>Associative</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kind</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role Category</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td>Valid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Valid</td>
</tr>
<tr>
<td>Associative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Valid</td>
</tr>
</tbody>
</table>

Two stereotypes can apply to whole/part associations: aggregation and composition.

7.1.2.1.2.3.1 Aggregation Association

An aggregation is a whole/part relationship which is sharable.

How to model an aggregation:

- It should be depicted by a dashed line
- It can have any of the four valid cardinalities on either the “whole” side or the “part” side (i.e., 1..*, 0..*, 1..1, 0..1) or in rare scenarios the cardinality may reflect specific numbers (as discussed previously)
- In UML, the connecting line at the “whole” end of the association should terminate with an open diamond.

To draw an aggregation association from Class A (the whole) to Class B (the part), select the aggregation icon from the Palette toolbar of PD. Then click on Class A and hold the mouse button...
down as you draw a line to Class B. An appropriate connector will appear when you release the mouse button.

Figure 7-20 shows an example of an aggregation association, depicting the relationship between the Engineer part class and the Engineering Team whole class. The whole/part association is sharable (aggregation) because an engineer can simultaneously be part of any number of engineering teams. The Engineering Team class is optional, because engineers can exist without belonging to an engineering team. The Engineer class is shown as mandatory with a cardinality of 2..* because the business requires an engineering team to have at least two members.

![Engineering Team Aggregation Diagram](image)

**Figure 7-20 Engineering Team Aggregation**

### 7.1.2.1.2.3.2 Composition Association

A *composition* is a whole/part relationship which is non-sharable.

How to model a composition:

- It should be depicted by a dashed line
- It must have a maximum cardinality of 1 on the “whole” side of the relationship (i.e., 0..1, 1..1)
- It can have any of the four valid cardinalities on the “part” side (i.e., 1..*, 0..*, 1..1, 0..1) or in rare scenarios the cardinality may reflect specific numbers (as discussed previously)
- The connecting line at the “whole” end of the association should terminate with a solid diamond.

**PD Guidance 76**

To draw a composition association from Class A (the whole) to Class B (the part), select the (composition) icon from the *Palette* toolbar of PD. Then click on Class A and hold the mouse button...
down as you draw a line to Class B. An appropriate connector will appear when you release the mouse button.

Figure 7-21 shows examples of composition associations, partially depicting the construction of an automobile. The three whole/part associations are non-sharable (compositions) because no instance of any of the parts can simultaneously be parts of two automobiles. In each case, the Automobile (the whole class in these associations) is optional, because engines, tires, and radios can exist as spare parts without belonging to an automobile. The Engine and Tire part classes are shown as mandatory because they are required parts for a proper automobile. The cardinality of Tire is shown as 4..5 to allow for an optional spare tire. Radio, of course, is shown as optional.

![Figure 7-21 Automobile Composition](image)

### 7.1.2.1.2.4 Additional Whole/Part Association Properties

The Whole/Part Association concept has the following additional properties:

*Whole/Part Association Property: is essential (optional)*

The “essential” concept was described above on page 79.

**PD Guidance 77**

To set the whole/part association “essential” property to true, open the association’s Association Properties worksheet. On the Rules tab, select the ![Add Objects](image) icon. A Selection worksheet will open, containing a list of Business Rules associated with the model. Check the selection box next to the Essential business rule.
Whole/Part Association Property: is immutable (optional)

The “immutable” concept was described above on page 79.

PD Guidance 78

To set the whole/part association “immutable” property to true, open the association’s Association Properties worksheet. On the Rules tab, select the (Add Objects) icon. A Selection worksheet will open, containing a list of Business Rules associated with the model. Check the selection box next to the Immutable business rule.

Whole/Part Association Property: is inseparable (optional)

The “inseparable” concept was described above on page 79.

PD Guidance 79

To set the whole/part association “inseparable” property to true, open the association’s Association Properties worksheet. On the Rules tab, select the (Add Objects) icon. A Selection worksheet will open, containing a list of Business Rules associated with the model. Check the selection box next to the Inseparable business rule.

7.1.2.1.2.5 Association Relationship Properties

The Association concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). One of the common properties, the “name”, requires the following additional explanation

Association Property: Name (mandatory)

The “name” indicates the reason for joining two classes in an association relationship and the name should be as descriptive of this reason as possible.

PD Guidance 80

The association “Name” property can be found in the General tab, Name field of the Association Properties Worksheet.

The following are some additional guidelines for naming associations:

- All associations (i.e., dependency, non-dependency, aggregation, and composition associations) must be named.

- In general, the association name should avoid the use of the word “is”. For example, use “employed by” rather than “is employed by”, since the word “is” can be inferred without its use and absence of this word will lessen the number of words displayed on the data model.
In general, the association name should avoid the use of the word “may”. For example, use “describes” rather than “may describe”, since the use of “may” implies an optional association, which should already be indicated in the association’s “minimum instances” property (i.e., by being set to zero (0), which means the association is optional).

Every association name should be a verb or verb phrase, such as “places”, “uses”, “contains”, “works for”, “initiates”, “categorizes”, etc. (as opposed to a noun).

The verb or verb phrase for each association name should be in the “active” form, where the subject of the sentence performs the action of the verb. For example, “each company enforces policies”; “each doctor authorizes prescription”. With passive verbs, the subject of the sentence receives the action of the verb. A passive verb is often preceded by a helping verb, such as “is”, “are”, or “was” or is followed with the word “by”. For example, “each research project is performed by scientists”. Active verb phrases are generally more direct, more clear, and shorter in length than passive ones.

All association names should be displayed in lower case form.

For dependency associations, the association should be named in the direction from the class upon which the other class is existentially dependent, to the dependent class.

Note: The direction in which the association is named should be indicated with the arrow head in the middle of the association line.

For non-dependency associations, the association should be named in the direction from the class where the cardinality’s maximum instances is one (1) to the class where the maximum instances is many (*), if the association’s cardinality is defined this way.

Note: A best-practice strategy for devising “active” verb phrases is to name them in this “one-to-many” direction, if the association’s cardinality is defined this way. If the association’s cardinality is not defined this way, the association may be named in either direction.

For aggregation and composition associations, the association should be named in the direction from the whole class to the part class. Generally, composition and aggregation associations will be given a name such as “contains”, “comprises”, or “consists of”.

Each association name should be positioned near the center of the horizontal or vertical association line, but the name should not cover up the arrow head in the middle of the association line. See Figure 7-22.
Figure 7-22 Example Showing Association Name

For the association between Doctor and Prescription, knowing that the convention is to read associations in the direction of the arrow in the middle of the association line, the reader can know that the proper way to read the association is by starting with the class on the left (Doctor), then read the association name (“authorizes”), the cardinality (“zero to many”) and then the class on the right (Prescription). Thus, this association would be read as:

*Each Doctor authorizes zero to many Prescriptions.*

- For “dependency” and “non-dependency” associations, if one wants to read an association in the opposite direction in which it is named, the inverse of the association name should be used (even though this name is not shown or even included as part of the definition of the association). For example, the association in Figure 7-22 between Prescription and Doctor could be read as “Each Prescription is authorized by one and only one Doctor”.

- An exception may be made for dependency associations to <<associative>> classes, when there are only two associations. Often such an <<associative>> has been created to resolve a many-to-many relationship between two classes. In such cases, it may be more descriptive to retain the name (and its inverse) that would have been applied to the many-to-many relationship. It is left to the modeler to decide whether the <<associative>> class has sufficient meaning on its own to be included in the name of the association relationship. Consider the example in Figure 7-23. The many-to-many relationship between the role Building Owner and the class Building can be well-named as “owns”.

Figure 7-23 Sample Many-to-Many Association

- Figure 7-24 resolves the many-to-many association by replacing it with the associative Building Ownership. The new associations to Building Ownership could both be named “has”. The semantics remains clear. The two association names can be read as:
A Building Owner has one to many Building Ownerships (of a Building), and

A Building has one to many Building Ownerships (by a Building Owner).

Figure 7-24 Naming Associations to an Associative Class

- Figure 7-25 presents the alternative naming convention for a pair of dependency associations to <<associative>> classes. Here, the names (“has”) have been replaced by the original name (“owns”) from the many-to-many relationship and its inverse (“owned by”). The modeler decided that the <<associative>> Building Ownership is not significant to the business and the original names of the associations are more descriptive. The two association names can be read as:

A Building Owner owns one to many Buildings, and

A Building is owned by one to many Building Owners.

Figure 7-25 Alternative for Naming Associations to an Associative Class

The above naming guidelines are summarized in Table 7-8.
Table 7-8 Summary of Naming Guidelines for the Four Stereotypes of Associations

<table>
<thead>
<tr>
<th>Association Stereotype</th>
<th>Name Required</th>
<th>Other Guideline</th>
<th>Active Verb Preferred</th>
<th>Direction to Name the Association</th>
<th>Lower Case Name</th>
<th>How to Read Association</th>
<th>Arrow Head in Middle of Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependency</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (i.e., subject performs the action)</td>
<td>To the side which is existentially dependent on the other side</td>
<td>Yes</td>
<td>In the direction of the arrowhead. “Each class A &lt;association name&gt; &lt;cardinality&gt; class B”</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>From the side where max # of instances is 1 to the side where the max # of instances is * (if this is the case); otherwise, can name in either direction</td>
<td>No</td>
<td>From the whole to the part. “Each (whole) class A &lt;association name&gt; &lt;cardinality&gt; (part) class B”</td>
<td>No</td>
</tr>
<tr>
<td>Aggregation</td>
<td></td>
<td></td>
<td></td>
<td>From the whole to the part</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Composition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

The Association concept also has the following properties, which are based on relationships to other concepts in the ELDM:

**Association Property: Generalized as association (optional)**

The association’s “generalized as association” property is an association that is more abstract than the association with this property. For example, as shown in Figure 7-26, the association between **Government Account Holder** and **IMF Financial Account** could have a more abstract association in the association between **Account Holder** and **Financial Account**. The more abstract association between **Account Holder** and **Financial Account** may be used by certain parts of the enterprise that do not require the more detailed associations. Typically each part of the enterprise
will either use the more abstract association or one or more detailed ones, but not associations at both levels.

PD Guidance 81

To set an association’s “generalized as association” property, use the Association worksheet, General tab, Other section, and select the appropriate (more abstract) association from the pull-down list of the Specializes field.

Figure 7-26 Example of Specialized and Generalized Associations

*Association Property: Specialized as association(s) (optional)*

The association’s “specialized as association(s)” property is a list of associations for which this association is a more abstract (or more generalized) representation. For example, as shown in Figure 7-26, the association between *Account Holder* and *Financial Account* could have two more specialized associations:

- The association between *Government Account Holder* and *IMF Financial Account* and
- The association between *Financial Institution* and *Financial Institution Account*.

The specialized associations may be used by certain parts of the enterprise that require the more detailed association. Typically each part of the enterprise will either use the more abstract form or more detailed form of the association, but not both.

PD Guidance 82

The “specialized as association(s)” property of an association can be found in the Association Properties worksheet, Dependencies tab, Extended Inverse Collections Sub-tab. The list of associations for which this association is a more abstract representation is the set of associations in the Collection Owner column which have a corresponding entry of “specialization” in the Inverse Collection Name column. PD
Association Ends

Each association has two association ends. Each association end is defined by the class next to it and has following two properties (which together are referred to as the association end’s “cardinality”):

**Association End Property: Minimum instances (mandatory)**

An association end’s “minimum instances” property is the least possible number of instances of the class that must exist for an instance of the class on the other end of the association. In most cases, this “minimum instances” property will be either zero (0) or one (1), however, in some circumstances, this value can be a number greater than one (e.g., 2, 3, etc.). For example, when modeling a geometric object such as a line, the line must have a minimum of two points.

If the “minimum instances” property is one (1), this indicates that an instance of the class is *mandatory* (i.e., must exist) for an instance of the class on the other end of the association.

If the “minimum instances” property is zero (0), this indicates that an instance of the class is *optional* (i.e., does not have to exist) for an instance of the class on the other end of the association.

**Note:** Optional associations are only allowed in those cases where even with perfect knowledge the association may not exist, as explained in APPENDIX G (Additional ELDM Guidelines; “Perfect Knowledge” pattern). Thus, for example, if the business knew that being a “Doctor” required a certification from a licensing organization, the “minimum instances” property on the association end next to Doctor License should be one (1), indicating the association is mandatory, despite the fact that for a particular doctor, facts about the license may not be obtained.

PD Guidance 83

“Minimum Instances” for the association end next to class A {or B} can be found in the Association Properties worksheet, Detail tab, Class A {or B} section, Multiplicity field, and set or modified using the pull-down list. “Minimum Instances” is the first of the two numbers.

**Association End Property: Maximum instances (mandatory)**

An association end’s “maximum instances” property is the greatest number of instances of the class that may exist for an instance of the class on the other end of the association. In most cases, this “maximum instances” property will be either one (1) or many (*), however, in some circumstances, this value can be a specific number greater than one (e.g., 2, 3, etc.). For example, when modeling geometric objects such as a triangle, the triangle can have a maximum of three points.

PD Guidance 84

“Maximum Instances” for the association end next to class A {or B} can be found in the Association Properties worksheet, Detail tab, Class A {or B} section, Multiplicity field, and set or modified using the pull-down list. “Maximum Instances” is the second of the two numbers.
Properties worksheet, Detail tab, Class A {or B} section, Multiplicity field, and set or modified using the pull-down list. “Maximum Instances” is the second of the two numbers.

Since an association end’s “minimum instances” property will typically be zero or one and an association end’s “maximum instances” property will typically be one or many, an association end’s cardinality will typically be one of the four combinations shown in Table 7-9. In rare circumstances, as described above, additional combinations are possible, since the “minimum instances” and “maximum instances” properties could also be specific numbers. Table 7-9 does not reflect these rarer scenarios.

Note: This table shows ERD cardinality notation for dependency and non-dependency associations. For whole/part associations, no cardinality notation is specified.

Table 7-9 Typical Cardinality Notations

<table>
<thead>
<tr>
<th>Typical Cardinality Notations for an Association End</th>
<th>Minimum number of instances (zero or one)</th>
<th>Maximum number of instances (one or many)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERD or UML 0..1</td>
<td>One (mandatory)</td>
<td>One</td>
</tr>
<tr>
<td>ERD or UML 1..1</td>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>ERD or UML 0..1</td>
<td>Zero (optional)</td>
<td>One</td>
</tr>
<tr>
<td>ERD or UML 1..*</td>
<td>One (mandatory)</td>
<td>Many</td>
</tr>
<tr>
<td>ERD or UML 0..*</td>
<td>Zero (optional)</td>
<td>Many</td>
</tr>
</tbody>
</table>

PD Guidance 85
PD does not allow the double vertical line for ERD indicated in Table 7-9 for 1..1 associations. A single vertical line is used instead. Similarly, a single circle will be used for 0..1 associations, rather than a circle with a vertical line beside it.

Note: A many-to-many association (i.e., an association with the cardinality on both ends of the association being “many”) is allowed in an ELDM. An associative class should be added between the two classes in a many-to-many association if the associative class has one or more attributes or if there are
one or more associations to the associative class (i.e., other than the associations that caused the associative class to be created). In many cases there is a business fact that belongs in this associative class (e.g., the date or reason the two classes became related to one another).

The Association End concept also has the following property, which is based on a relationship to another concept in the ELDM:

**Association End Property: Defined by class (mandatory)**

The association end’s “defined by class” property is the class that the association end belongs to and is closest to.

**PD Guidance 86**
The “defined by class” property of the A (or B) association end is the class depicted on the Class A (or B) side of the mini-diagram at the top of the Association Properties worksheet.

### 7.1.3 Relationship Properties Based on Relationships to other Concepts

The Relationship concept has the following properties which are based on relationships to other concepts in the ELDM:

**Relationship Property: Owned by business area (mandatory)**

The relationship’s “owned by business area” property is the business area that owns this relationship (i.e., the business area that owns a class that the relationship primarily describes).

**PD Guidance 87**
The “owned by business area” property of a relationship can be found in the appropriate (Association Properties, Specialization Properties, or Realization Properties) worksheet, Dependencies tab, Diagram sub-tab, Parent column.

**Relationship Property: Appears on diagram(s) (mandatory)**

The relationship’s “appears on diagram(s)” property is a list of all of the diagrams on which the relationship appears. A relationship must appear on at least one diagram, but may appear on many.

**Note:** The data modeling tool being used will most likely automatically keep track of this mapping, so the data modeler should not have to manually keep track of this information.

**PD Guidance 88**
The “appears on diagram(s)” property of a relationship is the set of diagrams found in the appropriate (Association Properties, Specialization Properties, or Realization Properties) worksheet, Dependencies tab, Diagram sub-tab, Name column.

**Note:** There may be multiple diagrams with the same name in the Model; the Parent column identifies the package within which this diagram appears.
**Relationship Property: Uses metadata pattern(s) (optional)**

The relationship’s “uses metadata pattern(s)” property is a list of the metadata patterns that the relationship uses. APPENDIX G (Additional ELDM Guidelines) describes a “Broad Application of Metadata” pattern, which explains the “Metadata Pattern” concept and how it can be used by an attribute.

---

**PD Guidance 89**

Metadata patterns are implemented using PD Business Rules.

The list of metadata patterns that the relationship uses can be found in the Association Properties, Specialization Properties, or Realization Properties worksheet, Rules tab, Name column.

See “Broad Application of Metadata” Pattern” on page G-4.
8 Data Type Concept

A data type is the format in which an attribute’s values may be represented. There are five kinds of data types: a primitive data type, domain, enumerated list, data structure, and a union. One of these five data types must be used for every attribute.

8.1 Data Type Properties

Most of the Data Type Properties are found in one of the tabs of the appropriate worksheet (Domain Properties, Struct Properties, Enum Properties, or Union Properties), which can be viewed by double clicking the Data Type in the browser or on a diagram. The appropriate tabs and fields are described later in this section or in APPENDIX B for the “Common Properties”.

Normally, you should view the appropriate data type properties worksheet with your “Favorite tabs” available; in other words, when viewing the properties worksheet, the word “More” should appear in the box in the lower left corner of the worksheet. If “Less” is displayed, click on the box to toggle the setting.

For ELDM work, the following tabs are the recommended favorites for the Domain Properties Worksheet: General, Also Known As, Extended Restrictions, Rules, Requirements, Dependencies, and Version Info. The following tabs are the recommended favorites for the Enum Properties Worksheets: General, Attributes, Also Known As, Extended Restrictions, Rules, Requirements, Dependencies, and Version Info. The following tabs are the recommended favorites for the Struct Properties Worksheets: General, Attributes, Also Known As, Rules, Requirements, Dependencies, and Version Info. The following tabs are the recommended favorites for the Union Properties Worksheets: General, Attributes, Also Known As, Rules, Requirements, Dependencies, and Version Info. To customize your favorite tabs, click on the Property Sheet Menu button to the right of the More (Less) box and select “Customize Favorite Tabs”. Ensure that each of the recommended favorites is checked. Unless you regularly work with PDMs or Requirement Models, you should uncheck the other options.

Note that the favorite tabs customization does not work as desired for all concepts. Some tabs, such as the Name Encoding and Physical Encoding tabs cannot be effectively removed from your favorites for certain worksheets. They are not needed for ELDM work, but will appear nevertheless. Also, some tabs that are required for ELDM work cannot be effectively added to your favorites for certain worksheets. In those cases, you will have to select More for that worksheet.

Note also that this customization applies to the way models are viewed at the workstation you are currently using. It does not change the model itself.

Figure 8-1, taken from Appendix A (ELDM Tool-Independent Metamodel), depicts the Data Type concept and its properties and relationships to other concepts.
The Data Type concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties).

8.1.1 Additional Data Type Property

The Data Type concept has the following additional property:

**Data Type Property: Type (mandatory)**

The data type’s “type” property indicates that the data type is one of five kinds, as previously described.

PD Guidance 91

Each data type is automatically assigned the appropriate type by PD, when the data type is created. The data type also appears in the appropriate Properties worksheet, General tab, Stereotype field.

8.1.2 Data Type Properties Based on Relationships to other Concepts

The Data Type concept also has the following properties, which are based on relationships to other concepts in the ELDM:
Data Type Property: Owned by business area (mandatory)

The data type’s “owned by business area” property is the business area that owns the data type (i.e., the business area within which the data type is “homed”).

PD Guidance 92
The “owned by business area” property of a data type can be determined from its placement within the PD browser, and the absence of a shortcut symbol ( ). (For example, indicates an <<enum>> data type owned by the business area, while indicates a shortcut to an <<enum>> data type owned by a different business area.)

Data Type Property: Describes attribute(s) (optional)

The data type’s “describes attribute(s)” property is a list of all the attributes to which the data type applies.

PD Guidance 93
The “describes attribute(s)” property of a data type can be found in the appropriate (Domain Properties, Struct Properties, Enum Properties or Union Properties) worksheet, Dependencies tab, Object Using the Class as Type sub-tab, Name column.
(Note that the class containing each attribute is found in the Parent column.)

8.2 Kinds of Data Types
The five kinds of data types are described in the following sections.

8.2.1 Primitive Data Type

A primitive data type is the fundamental data type or format in which an attribute’s values can be represented in order to be meaningful to the user. For example, the attribute begin date would be expected to have the primitive data type “date”; and a person’s name would be expected to have the primitive data type “string”.

A primitive data type (or a data structure) must be specified for every attribute in an ELDM.

8.2.1.1 Primitive Data Type Properties

PD Guidance 94
Primitive data types have been defined appropriately in the Language File. It is not expected that users will define their own primitive data types.

Figure 8-2, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Primitive Data Type concept and its property and relationships to other concepts.
8.2.1.1.1 Common Primitive Data Type Properties

The Primitive Data Type concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). One of the common properties for a Primitive Data Type, the “name” requires the following additional explanation.

Primitive Data Type Property: Name

The primitive data type name should be specified in lower camel case (e.g., integer, dateTime). The name will have one of five values for numeric primitive data types (date, dateTime, float, integer, or time) or will have one of three values for non-numeric primitive data types (boolean, string, or octets).

8.2.1.1.2 Additional Primitive Data Type Property

The Primitive Data Type concept has the following additional property:
**Primitive Data Type Property: Type (mandatory)**

The primitive data type’s “type” property indicates that the primitive data type is one of two types:

1. **Numeric** – a primitive data type that has a numeric format (i.e., float, integer, date, time, or dateTime primitive data types) or

2. **Non-numeric** – a primitive data type whose format is not numeric (i.e., string, boolean, or octets primitive data types).

---

**8.2.1.1.3 Primitive Data Type Properties Based on Relationships to other Concepts**

The Primitive Data Type concept also has the following properties, which are based on relationships to other concepts in the ELDM:

**Primitive Data Type Property: Describes numeric domain(s) (optional)**

The primitive data type’s “describes numeric domain(s)” property is a list of the “numeric domains” that are in part defined by this primitive data type (i.e., the numeric domains that use this primitive data type but specify additional characteristics for the required format).

---

**PD Guidance 96**

To determine which numeric domains in the model are described by a particular primitive data type, select the model’s main (highest level) diagram in the browser. From the “Model” menu, select “Domain Objects”. The List of Domain Objects worksheet will appear. Make sure the [Include sub packages] icon is selected. Also ensure that the **Primitive** column is displayed.

If the **Primitive** column is not displayed, select the [Customize Columns and Filter] icon. The Customize Columns and Filter worksheet will appear. Select the box next to the word “primitive” in the **Column Heading** column.

In the List of Domain Objects worksheet, find all numeric domains listed in the **Name** column, where that row’s **Primitive** column contains the particular desired primitive data type.

**Primitive Data Type Property: Describes non-numeric domain(s) (optional)**

The primitive data type’s “describes non-numeric domain(s)” property is a list of the “non-numeric domains” that are in part defined by this primitive data type (i.e., the non-numeric domains that use this primitive data type but specify additional characteristics for the required format).
To determine which non-numeric domains in the model are described by a particular primitive data type, select the model’s main (highest level) diagram in the browser. From the “Model” menu, select “Domain Objects”. The List of Domain Objects worksheet will appear. Make sure the ("Include sub packages") icon is selected. Also ensure that the Primitive column is displayed.

If the Primitive column is not displayed, select the ("Customize Columns and Filter") icon. The Customize Columns and Filter worksheet will appear. Select the box next to the word “primitive” in the Column Heading column.

In the List of Domain Objects worksheet, find all non-numeric domains listed in the Name column, where that row’s Primitive column contains the particular desired primitive data type.

The list of valid primitive data types for an ELDM will be managed by a centralized ELDM management team. Currently, there are eight valid primitive data types (five numeric and three non-numeric), as shown in Table 8-1.

### Table 8-1 Primitive Data Types for an ELDM

<table>
<thead>
<tr>
<th>Primitive Data Type Name</th>
<th>Primitive Data Type Definition</th>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>A list of two mutually exclusive values that express the possible states of an attribute. Values typically indicate a condition such as on/off or true/false.</td>
<td>Non-numeric</td>
<td>true/false, yes/no, on/off</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If the attribute is specified to be optional, then technically, there are three possible values: true, false, or null. If the attribute is mandatory, then only true or false are valid values.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| date                     | A time period that starts at midnight of a specified day and lasts for 24 hours. Dates are represented according to ISO 8601, using the Gregorian calendar with the largest of the following units given first (from left to right):  
  - year (YYY; i.e., 0000 through 9999)  
  - month (MM; i.e., 01 through 12)  
  - day (DD; i.e., 01 through 31).  
  and can be specified with a subset of these possible values (e.g., YYYYMM). See the ISO standard for allowable options and varying degrees of accuracy. | Numeric | 20070501          |
| dateTime                 | A combination of date and time values representing a single instant in time. DateTimes are represented according to ISO 8601 – i.e., in its most general form as | Numeric | 20070501104520  
                          |                               |        | 2007-05-01T11:04:20Z |
### Primitive Data Type Name | Primitive Data Type Definition | Type | Examples
--- | --- | --- | ---
[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss][TZD]. | Note: “TZD” indicates “Time Zone Designator”, the standard single alphabetic character representation. See the ISO standard for allowable options and varying degrees of accuracy. | |  |
float | A number with a decimal that can be less than, equal to, or greater than zero.  
Note: Float has a limited number of significant digits, typically from 6 to 15, depending upon the physical environment and type of float selected in an implementation environment (i.e., single or double precision). | Numeric | 12.579, 1.2478, -693.7 |
integer | A number without a decimal that can be less than, equal to, or greater than zero. | Numeric | 0, 9, 56, 847, -5, -256 |
ocets | Data whose format is unspecified, can be of any length, and is measured in octets (i.e., in 8 bit increments).  
Note: The format may be text, formatted documents, images, executable files, etc. | Non-numeric | The meeting minutes are as follows...; picture file |
string | A series of characters, regardless of the number of characters (i.e., length) or even whether the length is fixed or variable.  
Note: The allowed character set is not specified in an ELDM. A PDM for a specific implementation environment could specify the allowed character set. | Non-numeric | John, Smith, password1**, etc. |
time | A number to represent an instant in time. Time formats are represented according to ISO 8601, with the largest of the following units given first (from left to right):  
- hour (HH; i.e., 00 through 24)  
- minute (MM; i.e., 00 through 59)  
- second (SS; i.e., 00 through 60)  
- decimal fraction of a second (one or more integers; e.g., 0 through 999)  
- TZD (time zone designator; i.e., “Z” to indicate the Universal Coordinated Time (UTC), or +HH:MM or −HH:MM to indicate that the time is expressed in local time with a time zone offset in hours and minutes ahead of UTC if “+” or behind UTC if “−”) and can be specified with a subset of these possible values (e.g., HHMMSS). | Numeric | 104520 |
**Note:** One of the criteria used for determining whether a data format should be included as a valid primitive data type in the table above was whether the format would be meaningful to and understood by most users. Therefore, although it might be possible to represent some attributes as a hexadecimal coding scheme, this would severely limit its usefulness to users. Therefore, hexadecimal is not a valid primitive data type for an ELDM.

It is expected that the mapping of primitive data types and domains (discussed below) used in an ELDM to those used in a PDM designed for XSD generation, would be 1:1, for the most part. The mapping, however, may be one-to-many between an ELDM and PDMs designed for other environments, such as for Java code, C++ code, Oracle database schema, Sybase database schema, etc.

### 8.2.2 Domain Data Type

A *domain* data type is a re-usable description of the acceptable format for the value of an attribute that represents greater specificity than a primitive data type alone. Table 8-2 provides some examples of domains and shows how domains add precision to primitive data types.

**Table 8-2 Examples of Domains**

<table>
<thead>
<tr>
<th>Domain Name</th>
<th>Domain Properties</th>
<th>Associated Primitive Data Type</th>
<th>Sample Attribute (to which the domain may be applied)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Float</td>
<td>minimum inclusive value = 0</td>
<td>float</td>
<td>score</td>
</tr>
<tr>
<td>Angle</td>
<td>minimum inclusive value = 0; maximum exclusive value = 360</td>
<td>float</td>
<td>vehicle course direction</td>
</tr>
<tr>
<td>Rating 0-10</td>
<td>minimum inclusive value = 0; maximum inclusive value = 10</td>
<td>integer</td>
<td>confidence rating</td>
</tr>
<tr>
<td>Uuid</td>
<td>definition = an identifier standard used in software construction composed of a 16-byte number, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE).</td>
<td>string</td>
<td>id</td>
</tr>
<tr>
<td>Money</td>
<td>definition = an amount associated with the exchange of goods and services; precision = 2</td>
<td>float</td>
<td>dollar amount</td>
</tr>
<tr>
<td>US Zip Code</td>
<td>pattern = ^\d{5}$</td>
<td>string</td>
<td>zip code</td>
</tr>
</tbody>
</table>

As indicated above, a domain may be reused. Thus, as an example, the “uuid” domain in the table above may be used by multiple attributes. This re-usability feature simplifies the specification of the format allowed for attributes in that all of the domain properties do not have to be individually set every time they apply to an attribute; an attribute only needs to include the domain in its specification and all of the domain’s properties are included in the attribute’s specification.
8.2.2.1 Domain Data Type Properties

Figure 8-3, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Domain concept and its properties and relationships to other concepts.

8.2.2.1.1 Common Domain Data Type Properties

The Domain concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). Two of the common properties for a Domain, the “name” and “definition”, require the following additional explanation.

Domain Data Type Property: Name

The domain name should be specified in Title case with spaces between each word, if appropriate, and should describe the concept of the domain (e.g., Positive Integer). The name need not include the word “Domain”. Note: when displayed on diagrams (i.e., next to an attribute), the domain name should be shown in all lower case with spaces between each word.
The “Name” property of a domain appears in the Domain Properties worksheet, General tab, Name field.

**Domain Data Type Property: Definition**

In essence, the domain “definition” should describe the set of possible values for an attribute. The other properties of a domain, when used together, do the same thing as the definition, but in a more precise and unambiguous way.

The Definition section, of a domain appears in the Domain Properties worksheet, General tab, Definition section, Definition field.

### 8.2.2.1.2 Additional Domain Data Type Properties

The Domain concept has the following additional properties:

**Domain Data Type Property: Type (mandatory)**

The domain’s “type” property indicates that the domain is one of two types:

- Numeric Domain – a domain specification that applies to only numeric primitive data types (i.e., to float, integer, date, time, or dateTime primitive data types); or
- Non-numeric Domain – a domain specification that applies to only non-numeric primitive data types (i.e., to string, Boolean, or octets primitive data types).

The rest of the properties of a domain depend upon the domain type, as follows:

**Numeric Domains** may specify the following properties for any numeric primitive data types (i.e., for float, integer, date, time, or dateTime primitive data types):

**Numeric Domain Property: Precision (optional; only applicable for domains that apply greater specificity to float, time, or dateTime primitive data types)**

The numeric domain’s “precision” property is the maximum number of digits to the right of the decimal that may be specified for a “float” value (primitive data type) or the maximum number of digits to the right of the decimal for the second’s portion of the “time” or “dateTime” value. For example, “3” would indicate that the float number could be specified with up to three digits to the right of the decimal.
PD Guidance 101

The “Precision” property of a domain can be set by entering a non-negative integer in the Domain properties worksheet, Extended Restrictions tab, Numeric Type Restriction section, Precision field.

**Numeric Domain Property: Minimum Inclusive Value (optional)**

The numeric domain’s “minimum inclusive value” property is the lowest possible numeric value allowed for any numeric attribute specified by the domain. For example, if the attribute “percent” has a “Percentage Type” domain defined with a “minimum inclusive value” of 1, then the lowest possible value allowed would be 1.

**Note:** This domain property is normally only used by attributes defined with the “float” primitive data type, but technically could be used by any numeric primitive data type (integer, date, time, dateTime).

PD Guidance 102

The “Minimum Inclusive Value” property of a domain can be set by entering any value appropriate to the specified numeric primitive data type in the Domain properties worksheet, Extended Restrictions tab, Numeric Type Restriction section, Minimum Inclusive Value field.

**Numeric Domain Property: Maximum Inclusive Value (optional)**

The numeric domain’s “maximum inclusive value” property is the highest possible numeric value allowed for any numeric attribute specified by the domain. For example, if the attribute “percent” has a “Percentage Type” domain defined with a “maximum inclusive value” of 100, then the highest possible value allowed would be 100.

**Note:** This domain property is normally only used by attributes defined with the “float” primitive data type, but technically could be used by any numeric primitive data type (integer, date, time, dateTime).

PD Guidance 103

The “Maximum Inclusive Value” property of a domain can be set by entering any value appropriate to the specified numeric primitive data type in the Domain properties worksheet, Extended Restrictions tab, Numeric Type Restriction section, Maximum Inclusive Value field.

**Numeric Domain Property: Minimum Exclusive Value (optional)**

The numeric domain’s “minimum exclusive value” property is the numeric value that must be exceeded for any numeric attribute specified by the domain. For example, if the attribute “percent” has a “Percentage Type” domain defined with a “minimum exclusive value” of 1, then the lowest possible value allowed would have to be greater than 1 (e.g., 1.000001 would be OK, but 1.0 would not).
**Note**: This domain property is normally only used by attributes defined with the “float” primitive data type, but technically could be used by any numeric primitive data type (integer, date, time, dateTime).

**PD Guidance 104**

The “Minimum Exclusive Value” property of a domain can be set by entering any value appropriate to the specified numeric primitive data type in the **Domain properties** worksheet, **Extended Restrictions** tab, **Numeric Type Restriction** section, **Minimum Exclusive Value** field.

**Numeric Domain Property: Maximum Exclusive Value (optional)**

The numeric domain’s “maximum exclusive value” property is the numeric value that must be greater than any value allowed for a numeric attribute specified by the domain. For example, if the attribute **angle** has an “Angle Type” domain defined with a “maximum exclusive value” of 360, then the greatest possible value allowed would have to be less than 360 (e.g., 359.9999 would be OK, but 360 would not).

**Note**: This domain property is normally only used by attributes defined with the “float” primitive data type, but technically could be used by any numeric primitive data type (integer, date, time, dateTime).

**PD Guidance 105**

The “Maximum Exclusive Value” property of a domain can be set by entering any value appropriate to the specified numeric primitive data type in the **Domain properties** worksheet, **Extended Restrictions** tab, **Numeric Type Restriction** section, **Maximum Exclusive Value** field.

**Non-numeric Domains** may specify the following properties for any non-numeric primitive data types (i.e., for string, Boolean, or octets primitive data types):

**Non-numeric Domain Property: Pattern (optional)**

The non-numeric domain’s “pattern” property is a specification of the allowed character values for each individual character in the value of any character attribute specified by the domain. The pattern should be specified using the “regular expression” (also called “regex”) notation. For example, the pattern for an “IPv4 Address Type” domain (e.g., 1.243.211.162) would be:

`^(24[0-7]|2[0-3][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|1)?[0-9][0-9]?)$` (3)

**Note**: This assumes that a complete IPv4 address is considered a single string (rather than four separate numeric attributes).

The pattern for a Social Security Number (SSN), e.g. 123-45-6789, is:

`^(?i000)[?i666](?i73[4-9])74\d(0-6)\d(2)7(0-6)d7(012)\(\-\)?(?i000)\d\d3(?i0000)\d\4$`

**Note**: This assumes the following rules for SSNs:
• Numbers from 001-01-0001 to 772-99-9999 are valid
• The first 3 digits cannot be 000
• The first 3 digits also cannot be 666 or 734-749
• The middle group of digits cannot be 00
• The last 4 digits cannot be 0000.

The rules for regular expressions can be found in many places, to include:
• Within the World Wide Web Consortium (W3C) XML Schema Specification (i.e., in Part II on Datatypes, which can be found on the internet at: http://www.w3.org/TR/2000/WD-xmlschema-2-2000407/).

\[PD\text{ Guidance 106}\]

The “Pattern” property can be set by entering a Regular Expression in the Domain properties worksheet, Extended Restrictions tab, Non-numeric Type Restriction section, Regular Expression field.

**Note:** Many XML Schema software tools provide a regular expression validation capability.

**Non-numeric Domain Property: Minimum Length (optional)**

The non-numeric domain’s “minimum length” property is the minimum number of characters allowed for an attribute value specified by the domain. For example, if the attribute country code had a domain associated with it that had a minimum length of “3”, then all country code values would have to be at least 3 characters long.

\[PD\text{ Guidance 107}\]

The “Minimum Length” property can be set by entering a positive integer in the Domain properties worksheet, Extended Restrictions tab, Non-numeric Type Restriction section, Minimum Length field.

**Non-numeric Domain Property: Maximum Length (optional)**

The non-numeric domain’s “maximum length” property is the maximum number of characters allowed for an attribute value specified by the domain. For example, if the attribute country code had a domain associated with it that had a maximum length of “4”, then all country code values could be no longer than 4 characters long.

\[PD\text{ Guidance 108}\]

The “Maximum Length” property can be set by entering a positive integer, greater than or equal to the Minimum Length” property (if used) in the Domain properties worksheet, Extended Restrictions tab, Non-numeric Type Restriction section, Maximum Length field.
8.2.2.1.3 Domain Data Type Properties Based on Relationships to other Concepts

The Domain concept also has the following properties, which are based on relationships to other concepts in the ELDM:

**Domain Data Type Property: Appears on diagram(s) (mandatory)**

The domain’s “appears on diagram(s)” property is a list of all of the diagrams on which the domain appears. A domain must appear on at least one diagram, but may appear on many.

**Note**: The data modeling tool being used will most likely automatically keep track of this mapping, so the data modeler should not have to manually keep track of this information.

---

**Domain Data Type Property: Described by numeric primitive data type (optional)**

The domain’s “described by numeric primitive data type” property is the “numeric primitive data type” that in part defines this domain.

**Note**: Every domain must be defined in part by a single primitive data type, either numeric or non-numeric. For example, as shown in Table 8-2 above, the “angle domain” is described in part by the “float” primitive data type.

**Domain Data Type Property: Described by non-numeric primitive data type (optional)**

The domain’s “described by non-numeric primitive data type” property is the “non-numeric primitive data type” that in part defines this domain.

**Note**: Every domain must be defined in part by a single primitive data type, either numeric or non-numeric. For example, as shown in Table 8-2 above, the “US zip code domain” is described in part by the “string” primitive data type.

---

**PD Guidance 110**

The “Described by primitive data type” property can be set by selecting a value from the Domain properties worksheet, Extended Restrictions tab, Base Primitive Type section, Base Primitive Type pull-down list.

How to model domains:

1. Create a class for the domain.
PD Guidance 111
Select the (domain) icon from the Base Extension toolbar of PD, and click within any diagram in the appropriate Business Area.

2. Assign the domain class to any attribute whose values are specified by the domain.

PD Guidance 112
See the “PD Guidance 55” write-up on page 57 regarding selecting a classifier.

8.2.3 Enumerated List Data Type

An enumerated list data type (i.e., an “enum”) is a re-usable list of allowed values for an attribute(s).

For example, the Skill Name Enum, shown in Figure 8-4, might specify: “active listening”, “critical thinking”, “negotiation”, and “effective verbal communication” as valid values. This enumerated list might apply to an attribute named skill in the class Person Skill.

![Skill Name Enum]

Figure 8-4 Sample Enumerated List

Enumerated lists may not inherit from one another. Instead, each enumerated list should be shown independently (i.e., not related to any other enumerated list). Thus, even if the allowed values for one enumerated list is a subset of the values of another one, a separate enumerated list should be created for each list.

8.2.3.1 Enumerated List Data Type Properties

Figure 8-5, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Enumerated List concept and its properties and relationships to other concepts.
8.2.3.1.1 Common Enumerated List Data Type Properties

The Enumerated List concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). One of the common properties for an Enumerated List, the “name”, requires the following additional explanation.

Enumerated List Data Type Property: Name

The name of an enumerated list should end with “Enum”.

PD Guidance 113
The “Name” property of an enumerated list appears in the Enum Properties worksheet, General tab, Other section, Name field.

8.2.3.1.2 Additional Enumerated List Data Type Properties

The Enumerated List concept has the following additional properties:
Enumerated List Data Type Property: Type (mandatory)

The enumerated list’s “type” property indicates that the enumerated list is one of two types:

1. **Internal Enumerated List** – an enumerated list that is maintained within an ELDM (i.e., the master list of valid entries is kept within the data model) to facilitate the management of the data model. SMEs may not even know (or even need to know) that these lists are maintained (e.g., Party Type, which consists of Person and Organization). An internal enumerated list should only be created if it has a minimal amount of metadata (not including the common properties identified in Appendix B (ELDM Common Properties)). Thus, if the enumerated list requires additional metadata, such as “begin effective date”, “end effective date”, “order”, etc., then an external enumerated list must be used.

2. **External Enumerated List** – an enumerated list that is maintained externally from an ELDM (i.e., the master list of valid entries is kept outside the data model). There are several reasons why an enumerated list should be maintained externally (and any one of the following reasons can justify making the enumerated list external):

   - The enumerated list values change frequently.
     
     **Note:** Keeping these enumerated lists external from the model simplifies the process for changing values, since the model does not necessarily need to be changed when values are changed.

   - The enumerated list requires more properties than the ones available for internal enumerated lists. For example, if the enumerated list requires “begin effective date”, “end effective date”, “order”, etc.

   - An organization outside of the enterprise controls the list of valid entries.
     
     **Note:** If either of the first two reasons apply to the enumerated list (i.e., if the list changes frequently or if the list has additional metadata), then an organization outside of the enterprise does not necessarily have to be the organization that controls the list of values; the centralized ELDM team could also be the organization that controls the list of values (and in fact, if the list of values for the external enumerated list are used by a partitioning attribute and reflect the names of the subclasses, then the centralized ELDM team will likely be the controlling organization or at least will exercise final “veto” rights on changes to the list of values).

---

**PD Guidance 114**

To set the enumerated list “type” property to external, open the enumerated list’s Enum Properties worksheet. On the Rules tab, select the ![Add Objects](image) (Add Objects) icon. A Selection worksheet will open, containing a list of Business Rules associated with the model. Check the selection box next to the External business rule. If that business rule is not selected, the type of the enumerated list is internal.
**Enumerated List Data Type Property: Rate of Change (mandatory)**

The enumerated list’s “rate of change” property indicates how frequently the set of enumerated list values are expected to be changed. Valid values are: frequently, rarely, or never, which are defined as follows:

- **Frequently** – one or more changes to the enumerated list (i.e., the addition of a new value, a change, or removal of an existing value) are expected to be made each year or even more frequently.
- **Rarely** – a change to the enumerated list is expected to be made less frequently than once per year (e.g., once every two years).
- **Never** – the enumerated list is not expected to ever change.

PD Guidance 115

The “Rate of Change” property of an Enumerated List can be found in the Enumerated List Properties worksheet, Extended Restrictions tab, Rate of Change section, Change field, and set using the pull-down list.

In addition, an internal enumerated list may specify the following property for each valid entry of the enumerated list:

**Enumerated List Data Type Property: Value(s) (mandatory)**

The internal enumerated list’s “value(s)” property is a list of actual valid, permissible values (i.e., a string value, a word or set of words, a code value, etc.). For example, the **Gender Type** internal enumerated list may have the following as valid values: “Male” and “Female”.

**Note:** The individual values should be formatted with upper and lower case, as appropriate, to be most meaningful for the user (e.g., DoD, USA, etc.). If the values represent the names of subclass classes, the values should match the names of the subclasses verbatim, to include following the same case convention for the subclass (e.g., Savings Account, Checking Account). Also, while some enumerated lists have multiple ways to express the values contained within the list (e.g., as a code value, a name, an abbreviated name, a value with spaces removed, etc.), the value captured here is to represent the one, common and sharable way that the enterprise desires to express the value. The data modeler should also be aware that some implementations have difficulties with a space between words in an enumerated list value.

PD Guidance 116

The “Value” properties of the valid entries of an enumerated list are found in the Enumerated List Properties worksheet, Attributes tab, Name column.
8.2.3.1.3 Enumerated List Data Type Properties Based on Relationships to other Concepts

The following property for an enumerated list is based on a relationship that an enumerated list has to another concept:

**Enumerated List Data Type Property: Appears on diagrams (s) (mandatory)**

The enumerated list’s “appears on diagram(s)” property is a list of all of the diagrams on which the enumerated list appears. An enumerated list must appear on at least one diagram, but may appear on many.

**Note:** The data modeling tool being used will most likely automatically keep track of this mapping, so the data modeler should not have to manually keep track of this information.

---

**PD Guidance 117**

The “appears on diagram(s)” property of an enumerated list is the set of diagrams found in the Enum Properties Worksheet, Dependencies Tab, Diagram sub-tab, Name column. (Note that there may be multiple diagrams with the same name in the Model; the **Parent** column identifies the package within which this diagram appears.)

---

**How to model enumerated lists:**

1. A class should always be created for an enumerated list.

   **PD Guidance 118**
   
   Select the ![Enum icon](image) from the **Base Extension** toolbar of PD, and click within the appropriate diagram.

2. Set the color of the enumerated list based on which business area owns the enumerated list.

   - If the enumerated list is owned by the business area of the diagram on which the enumerated list appears, its color should be light green
   - If the enumerated list is owned by a business area other than the business area of the diagram on which the enumerated list appears, its color should be pale red.

3. Show the values listed alphabetically as attributes or within a note, depending on whether the enumerated list is an internal or external one.

   - If the enumerated list is an internal one, show the complete list of values as attributes in the enumerated list class.
If the enumerated list is an external one, you may wish to create a note next to the enumerated list class showing the complete list of values, if this list is short enough to fit on the diagram. If the list is not short enough, you may show as many values on the diagram as practical and then add a row with "..." to indicate that there are more values but they are not being shown. This note is not required.

### PD Guidance 119

To create a note on a diagram, select the 📝 (note) icon from the Palette toolbar of PD, and click within the appropriate diagram.

If the enumerated list is an external one, add the “external” business rule, as explained in PD Guidance 114 on page 110.

4. Within the class that contains an attribute described by the enumerated list, make sure the name of the enumerated list is shown within the class as describing the attribute.

### PD Guidance 121

See the “PD Guidance 55” write-up on page 57 regarding selecting a classifier.

5. Place the enumerated list next to the class that contains an attribute described by the enumerated list (where there is room on the diagram to do so without overlapping another class, relationship, etc.).

Table 8-3 provides examples for how to graphically depict enumerated lists on diagrams depending on the owning business area and whether the enumerated list is internal or external.

**Table 8-3 Examples of How to Depict Enumerated Lists on Diagrams**

<table>
<thead>
<tr>
<th>Owning Business Area</th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&lt;&lt;enum&gt;&gt;</strong> Gender Enum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male : string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Female : string</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>&lt;&lt;enum&gt;&gt;</strong> Building Name Enum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{External}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Building Name Enum values:
* Headquarters Building
* Central Downtown Building
* East Side Downtown Building
* Main Street Building
Figure 8-6 shows the proper placement of an enumerated list next to the class that contains an attribute described by the enumerated list.

8.2.4 Data Structure Data Type

A data structure is a highly-cohesive set of two or more mandatory attributes that may be re-used within the data model to describe an attribute. These attributes represent an n-dimensional value space. For example, a “point” in two-dimensional Cartesian coordinate system consists of the values “x” and “y”. Data structures help to simplify models.

In order to include a data structure in an ELDM, a class should be created that ends with the word “Struct”. Further, this class should be given the stereotype “<<struct>>”.

For example, “color”, which represents the intersection of a three-dimensional system, consists of values for “hue”, “saturation”, and “intensity”. Color may be used in multiple places within a data model. All three of these attributes are necessary (i.e., mandatory) in order to specify a color. These three attributes represent a data structure (e.g., named “Color Struct”). Any other class (e.g., Vehicle) that needs to include a color (with these three attributes) would need have an attribute (e.g., color) that references the data structure (Color Struct), as shown in Figure 8-7.
Figure 8-7 Usage of a Data Structure

Note: Following the same color convention as with an enumerated list, if a data structure is owned by the business area of the diagram on which the data structure appears, the data structure class should be light green. If it is owned by a business area other than the business area of the diagram on which the data structure appears, it should be pale red.

8.2.4.1 Data Structure Data Type Properties:

Figure 8-8, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Data Structure concept and its properties and relationships to other concepts.

Figure 8-8 The Data Structure Concept and Its Properties and Relationships

8.2.4.1.1 Common Data Structure Properties

The Data Structure concept has a number of common properties as shown and defined in APPENDIX B (ELDM Common Properties). One of the common properties for a Data Structure, the “name”, requires the following additional explanation.
Data Structure Data Type Property: Name

As stated above, the name of a data structure should end with “Struct”.

PD Guidance 122
The “Name” property of a data structure appears in the Struct Properties worksheet, General tab, Name field.

8.2.4.1.2 Data Structure Data Type Properties Based on Relationships to other Concepts

The Data Structure concept also has the following properties, which are based on relationships to other concepts in the ELDM:

Data Structure Data Type Property: Consists of attributes (mandatory)

The data structure’s “consists of attributes” property is a list of the attributes that are included in the data structure (i.e., the attributes that are part of the n-dimensional value).

PD Guidance 123
The list of attributes of a data structure appears in the Struct Properties worksheet, Attributes tab, Name column.

Data Structure Data Type Property: Appears on diagram(s) (mandatory)

The data structure’s “appears on diagram(s)” property is a list of all of the diagrams on which the data structure appears. A data structure must appear on at least one diagram, but may appear on many.

Note: The data modeling tool being used will most likely automatically keep track of this mapping, so the data modeler should not have to manually keep track of this information.

PD Guidance 124
The “appears on diagram(s)” property of a data structure is the set of diagrams found in the Struct Properties Worksheet, Dependencies Tab, Diagram sub-tab, Name column.
(Note that there may be multiple diagrams with the same name in the Model; the Parent column identifies the package within which this diagram appears.)

How to model a data structure:

1. Create a class that ends with the word “Struct” and has the stereotype <<struct>>.

PD Guidance 125
Select the (struct) icon from the Base Extension toolbar of PD, and click within the appropriate diagram.

2. Set the color of the data structure based on which business area owns the data structure.
If the data structure is owned by the business area of the diagram on which the data structure appears, its color should be light green.

If the data structure is owned by a business area other than the business area of the diagram on which the data structure appears, its color should be pale red.

3. Within the class that contains an attribute described by the data structure, make sure the name of the data structure is shown within the class as describing the attribute.

4. Place the data structure next to the class that contains an attribute described by the data structure (where there is room on the diagram to do so without overlapping another class, relationship, etc.).

Figure 8-9 depicts a sample data structure.

```
<<struct>>
Color Struct
hue : ColorFloat
saturation : ColorFloat
intensity : ColorFloat
```

Figure 8-9 Sample Data Structure

8.2.5 Union Data Type

A union data type is a set of two or more data types, any of which may be used within the data model to describe an attribute.

In order to include a union in an ELDM, a class should be created whose name ends with the word “Union”. Further, this class should be given the stereotype “<<union>>”.

For example, if separate enumerations existed for the sets of countries on each populated continent (Asia Country Enum, Africa Country Enum, etc.), a World Country Union could be created consisting of those six individual enumerations.

8.2.5.1 Union Data Type Properties:

Figure 8-10, taken from APPENDIX A (ELDM Tool-Independent Metamodel), depicts the Union concept and its properties and relationships to other concepts.
8.2.5.1.1 Common Union Data Type Properties

The Union concept has a number of common properties as shown and defined in Appendix B (ELDM Common Properties). One of the common properties for a Union, the “name”, requires the following additional explanation.

*Union Data Type Property: Name*

As stated above, the name of a union should end with “Union”.

---

**PD Guidance 126**

The “Name” property of a union appears in the *Union Properties* worksheet, *General* tab, *Name* field.

8.2.5.1.2 Union Data Type Properties Based on Relationships to other Concepts

The Union concept also has the following properties, which are based on relationships to other concepts in the ELDM:

*Union Data Type Property: Consists of data types (mandatory)*

The union’s “consists of data types” property is a list of the data types that are included in the union.

---

**PD Guidance 127**

The list of data types of a union appears in the *Union Properties* worksheet, *Attributes* tab, *Data Type* column.
Union Data Type Property: Appears on diagram(s) (mandatory)

The union’s “appears on diagram(s)” property is a list of all of the diagrams on which the union appears. A union must appear on at least one diagram, but may appear on many.

Note: The data modeling tool being used will most likely automatically keep track of this mapping, so the data modeler should not have to manually keep track of this information.

PD Guidance 128
The “appears on diagram(s)” property of a union is the set of diagrams found in the Union Properties Worksheet, Dependencies Tab, Diagram sub-tab, Name column. (Note that there may be multiple diagrams with the same name in the Model; the Parent column identifies the package within which this diagram appears.)

How to model a union:

1. Create a class that ends with the word “Union” and has the stereotype <<union>>.

   PD Guidance 129
   Select the (union) icon from the Base Extension toolbar of PD, and click within the appropriate diagram.

2. Open the Union Properties Worksheet, Attributes tab.

3. For each data type which comprises the union, enter an arbitrary name (such as datatype1, datatype2, etc.) in the Name column of a row of the Attributes tab and enter the name of the data type on that same row, in the Data Type column.

4. Set the color of the union based on which business area owns the data structure.
   
   o If the union is owned by the business area of the diagram on which the data structure appears, its color should be light green
   
   o If the union is owned by a business area other than the business area of the diagram on which the union appears, its color should be pale red.

5. Within the class that contains an attribute described by the data structure, make sure the name of the union is shown within the class as describing the attribute.

6. Place the union next to the class that contains an attribute described by the data structure (where there is room on the diagram to do so without overlapping another class, relationship, etc.).

   Figure 8-11 depicts a sample union.
Figure 8-11 Example of a Union

<table>
<thead>
<tr>
<th>&lt;&lt;union&gt;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>World Country Union</td>
</tr>
<tr>
<td>datatype1 : africaCountryEnum</td>
</tr>
<tr>
<td>datatype2 : asiaCountryEnum</td>
</tr>
<tr>
<td>datatype3 : australiaCountryEnum</td>
</tr>
<tr>
<td>datatype4 : europeCountryEnum</td>
</tr>
<tr>
<td>datatype5 : northAmericaCountryEnum</td>
</tr>
<tr>
<td>datatype6 : southAmericaCountryEnum</td>
</tr>
</tbody>
</table>

World Country Union: Africa, Asia, Australia, Europe, North America, South America
APPENDIX A. ELD Tool-Independent Metamodel

The purpose of this appendix is to provide an overview of the concepts and properties needed to specify an ELDM independent of any tool used to capture this information.

Figure A-1 graphically depicts an ELDM tool-independent metamodel. This diagram shows:

- All of the required concepts for an ELDM
- All of the required relationships between the concepts
- All of the required properties, other than the common properties, for each concept
- All of the concepts to which the set of common properties apply, indicated with {Common} below the concept name in Figure A-1.

**Note:** The common properties are explained in APPENDIX B (ELDM Common Properties).

Regardless of the data modeling tool chosen to represent an ELDM, each of the above concepts, relationships, and properties are used to describe an ELDM.

**Note:** Each of the concepts and properties (other than the common properties) has already been defined throughout this document.
Figure A-1 Tool-Independent Metamodel for an ELDM
The purpose of this appendix is to describe common properties that apply to the concepts required for an ELDM. As can be seen in Figure B-1, there are many common properties required for describing the ELDM concepts.

This diagram depicts common properties that apply to the following ELDM concepts:
* Model
* Business Area
* Diagram
* Entity/Class
* Attribute
* Relationship
* Data Type
* Internal Enumerated List Valid Entry
* Metadata Pattern

**Figure B-1 Common Properties That Apply to ELDM Concepts**

Table B-1 shows the common properties that apply to specific ELDM concepts and indicates whether the property is mandatory ("M") or optional ("O") for that concept.
Table B-1 Common Properties Applied to ELDM Concepts

<table>
<thead>
<tr>
<th>ELDM Concepts</th>
<th>identifier</th>
<th>name</th>
<th>concept</th>
<th>security marking</th>
<th>definition</th>
<th>definition primary source</th>
<th>definition security marking</th>
<th>detail</th>
<th>note</th>
<th>note overall security marking</th>
<th>alternative term</th>
<th>creation user</th>
<th>creation user name</th>
<th>last modification dateTime</th>
<th>last modification user identifier</th>
<th>modification user name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Business Area</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Diagram</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Class</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Attribute</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Association</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Inheritance</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>DataType</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>Internal Enumerated List Valid Entry</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>O</td>
</tr>
<tr>
<td>Metadata Pattern</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>
The common properties shown in Table B-1 are defined as follows:

**Common Property: Identifier**

The “Identifier” property (applied to all of the concepts shown in Table B-1) is the unique identifier for the concept. Typically, this identifier will be a property controlled by the data modeling tool and will often not be visible to the user from within the tool.

PD Guidance 130
The “Identifier” property of a concept is maintained by PD and is not visible to the user.

**Common Property: Name**

The “name” property (applied to all of the concepts shown in Table B-1) is a word or words by which the concept is known. It is how one refers to the concept and should reflect the purpose for creating the concept. The name must comply with the ELDM Acronym Guidelines (APPENDIX D).

PD Guidance 131
The “Name” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, Name field.

**Common Property: Concept security marking**

The “concept security marking” property (applied to the concepts indicated in Table B-1) is defined in a separate document, the “ELDM Security Marking Guidelines.”

PD Guidance 132
The “Concept security marking” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, [Concept] security marking field. It can be set from the pull-down list provided.

**Common Property: Definition**

The “definition” property (applied to the concepts indicated in Table B-1) describes the concept which serves to differentiate it from related concepts. The definition communicates as precisely as possible what the concept means or represents. The definition is crucial for sharing and reuse of the data model as well as for validation of requirements. APPENDIX E (ELDM Definition Guidelines) provides additional guidelines for developing high quality definitions for classes and attributes.

Since the definition describes one enterprise concept, there should be no need for an “alternate” definition, in the way that a standard dictionary provides multiple meanings for a single word. However, the definition may be restated for clarification, using greater or lesser technical specification, as long as the restatement is consistent in meaning. For example, the concept “point” might have the following definition: “A particular spot, place or position that does not have a spatial
area, size, measure or direction; in more technical terms, a 0-dimensional geometric primitive representing a position, but not having extent.”

Examples should be put in the “note” property; not in the “definition” property.

PD Guidance 133

The “definition” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, Definition section, Definition field.

Note that for almost all concepts, the Definition field is not directly editable. To enter or modify a definition, click on the “Edit” button to the right of the Definition field. An “Edit Definition” window will open. Enter or edit the definition in this window, and click on the “OK” button. The new or modified definition will appear in the Definition field.

Note also that a number of “tokens” have been created, to simplify the entry and maintenance of definitions. The tokens and their usage are described in PD Guidance within APPENDIX E.

Common Property: Definition primary source

The “definition primary source” property specifies a citation for the main reference from which the knowledge to develop the definition was obtained or derived.

PD Guidance 134

The “definition primary source” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, Definition section, Source field. It can be set from the pull-down list provided.

Common Property: Definition source detail

The “definition source detail” property specifies additional information about the source(s) used to define a concept (i.e., beyond just the name of the primary source). This property provides a way for those who record definitions to communicate in a more detailed way exactly where the information was obtained to develop the definition. Examples of the types of additional information that could be provided include:

- The fact that the definition was derived from the primary source (e.g., “Derived from <short name of source>”)
- The section(s) from within the source that were used to obtain the definition (e.g., “Derived from the definitions for <term 1>, definition 3 and <term 2>, definition 5.”)
- Additional source(s) used (i.e., that, like the primary source, also accurately reflect the intended meaning of the concept).
Note: All source citations must be formatted as described in Appendix F (ELDM Definition Sourcing Guidelines).

- The web address and date retrieved if the publication was retrieved from the internet. For example, “Retrieved on October 14, 2008 from http://www.psychologytoday.org/journals/webref.htm.l”

PD Guidance 135
The “definition source detail” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, Definition section, Source Details field.

*Common Property: Definition security marking*

The “definition security marking” property (applied to the concepts indicated in Table B-1) is defined in a separate document, the “ELDM Security Marking Guidelines.”

PD Guidance 136
The “definition security marking” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, Definition section, Security Marking field. It can be set from the pull-down list provided.

*Common Property: Note*

The “note” property is a statement or comment about the concept that is useful but not essential to the definition of the concept. The note property may contain examples, non-essential characteristics, optional parts often associated with the concept, definitions of terms used within the concept definition or extra information that is useful to understand what the concept is. If the comment is essential to the definition, the comment belongs in the definition. The “note” property should be portion marked with a separate portion marking for each section.

It is a best practice to keep notes terse and to avoid the use of special characters in notes. Some modeling tools have problems with lengthy notes or notes containing special characters.

PD Guidance 137
The “note” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, Note section, Note field. (The Notes tab, a distinct tab from the General tab, is normally not displayed and is not used.)

*Common Property: Note security marking*

The “note overall security marking” property is the security marking to be applied to the note as a whole (i.e., the highest security marking of any of the portion markings found in the note), in the portion marking security marking format (e.g., U//FOUO). This property is mandatory if the “note” property is populated.
PD Guidance 138
The “note overall security marking” property of a concept is always found in that concept’s Concept Properties worksheet, General tab, Note section, Security Marking field. It can be set from the pull-down list provided.

**Common Property: Has alternative term (optional)**

The “has alternative term” property of a concept is a list of alternate (or alias) names that the business uses for the class along with the context or portion of the business where that name is used.

PD Guidance 139
The “has alternative term” property of a concept is always entered on that concept’s Concept Properties worksheet, Also Known As tab, aka field, and is entered as free formatted text listing class aliases along with business usage or other context.

**Common Property: Creation date time**

The “creation date time” property (applied to each of these concepts) is the date and time that the concept was created.

PD Guidance 140
The “creation date time” property of a concept is always found in that concept’s Concept Properties worksheet, Version info tab, Creation section, Date field. It is set automatically and cannot be altered by the user.

**Common Property: Creation user identifier**

The “Creation user identifier” property (applied to each of these concepts) is the user identifier of the individual who initially created the concept in the ELDM.

PD Guidance 141
The “creation user identifier” property of a concept is always found in that concept’s Concept Properties worksheet, Version info tab, Creation section, User field. It is set automatically and cannot be altered by the user.

**Common Property: Creation user name**

The “creation user name” property (applied to each of these concepts) is the user name of the individual who initially created the concept in the ELDM.

PD Guidance 142
The “creation user name” property of a concept does not appear in the current PD implementation.
**Common Property: Last modification date time**

The “last modification date time” property (applied to each of these concepts) is the most recent date and time that the concept was updated.

PD Guidance 143
- The “last modification date time” property of a concept is always found in that concept’s Concept Properties worksheet, Version info tab, Last modification section, Date field. It is set automatically and cannot be altered by the user.

**Common Property: Last modification user identifier**

The “last modification user identifier” property (applied to each of these concepts) is the user identifier of the individual who last made a change to the concept in the ELDM.

PD Guidance 144
- The “last modification user identifier” property of a concept is always found in that concept’s Concept Properties worksheet, Version info tab, Last modification section, User field. It is set automatically and cannot be altered by the user.

**Common Property: Last modification user name**

The “last modification user name” property (applied to each of these concepts) is the user name of the individual who last made a change to the concept in the ELDM.

PD Guidance 145
- The “last modification user name” property of a concept does not appear in the current PD implementation.
APPENDIX C.  ELDM NAMING GUIDELINES

The purpose of this appendix is to provide guidelines to data modelers for naming classes, domains, enumerated lists, data structures, and attributes in an ELDM. The essence of these guidelines is that the names of these concepts are to be clear and understandable to all members of the business.

There are two types of naming guidelines contained within this appendix: (1) standards, which are conventions that must be followed; and (2) best practices, which are highly recommended conventions (since they represent best practices), but they are optional.

This appendix organizes the naming guidelines into four sections, as follows:

1. Standards for Class, Domain, Enumerated List, Data Structure, and Union Names

2. Best Practices for Class, Domain, Enumerated List, Data Structure, and Union Names

3. Standards for Attribute Names contained in Classes or Data Structures

4. Best Practices for Attribute Names contained in Classes or Data Structures.

1. Standards for Class, Domain, Enumerated List, and Data Structure Names

   - **Uniqueness Standard.** Names for these concepts must be unique within the entire scope of an ELDM. Thus, for example, there should not be two separate classes, both named “Person”, with different meanings in an ELDM.

     PD Guidance 146

     PD enforces the Uniqueness Standard. If the modeler attempts to name a class with the same name as an existing class, the warning “That name already exists” will appear.

   - **Title-Case Standard.** The name must be in title case, except when acronyms are included. Thus, the following are acceptable names: “Person Role”, “SO Line Item”.

   - **Special-Characters Standard.** No characters other than letters, numbers, periods, or hyphens are allowed in the name. For example, the following characters are not allowed: slashes (/ or \), ampersands (&), asterisks (*), dollar signs ($), pound signs (#), apostrophes (‘), quotes (“”), underscores (_), etc.

   - **Spaces-Separator Standard.** Multiple terms in the name will be separated by spaces (e.g., “Person Role”; not “Person-Role” or “Person_Role”).

   - **Name-Restriction Standard.** Names for these concepts must not contain any of the following:

     o Words in possessive form (e.g., “Person’s Skill”)

     o Indefinite or definite articles (e.g., “a”, “an”, “the”)
2. **Best Practices**

- **Conjunctions**: (e.g., “and”, “but”, “or”)
- **Verbs**: (e.g., “Person Lives At Location”)
- **Prepositions**: (e.g., “from”, “at”, “by”, “of”, etc.)

- **Definition Standard**: The name for these concepts should be consistent with its definition. Thus, if one first reads the definition without looking at the name of the concept, one should not be surprised to see the name of the concept.

- **Singular-Noun Standard**: A name for these concepts must consist of a singular noun or noun phrase. For example, use “Order”, not “Orders”.

- **Spelled-Out-Name Standard**: Names for these concepts must consist of fully spelled-out word(s) and/or acronyms, as explained in APPENDIX D (ELDM Acronym Guidelines).

- **Abbreviation Standard**: Names for these concepts must not contain any abbreviations. An abbreviation is a shortened form of a word (e.g., “cnt” for “count”; “dept” for “department”; etc.). Abbreviations are often formed differently by different groups for the same word, thus leading to ambiguity as to what word the abbreviation is intended to represent.

### 2. Best Practices for Class, Domain, Enumerated List, and Data Structure Names

- **Describe-Business-Meaning Best Practice**: Each of these concepts should be given a name that represents its business meaning (i.e., a name that has meaning to the user community).

- **Use-of-Adjectives Best Practice**: Most names for these concepts should contain an adjective(s) to differentiate information about the basic concept, where appropriate (e.g., “Credit Risk”).

- **Descriptive Best Practice**: A balance must be sought between names for these concepts that are short and those that are longer and more descriptive. For example, some “independent” classes, such as Party, Report, etc. can be named well with a single word. However, most classes will require a longer name. Even if a group of business users use a single word, the data modeler should add modifiers to the name to distinguish between different “flavors” of the concept that may exist in other areas of the business (e.g., Government Organization, External Customer, Investigation Report, etc.).

- **Dependent-Class-Naming Best Practice**: A dependent class name should generally begin with the name of the class upon which it depends for its existence followed by one or more words to communicate what the dependent class describes about the other class (e.g., “Person Skill”).

- **Associative-Class-Naming Best Practice**: An associative class name should represent an association of the concepts of each of the classes upon which it depends for its existence. Associative classes are often best named by using all or parts of the names of the classes upon which it depends for its existence followed by a word to indicate the association of those
3. Standards for Attribute Names contained in Classes or Data Structures

- **Uniqueness Standard.** The attribute name must be unique among attributes of the class or data structure. Attributes of different classes (or of different data structures) can have the same name. For example, if the class **Benefit** has an attribute named “description”, there can be no other attributes within **Benefit** named “description”. However, there can be other classes, such as **Race**, that have an attribute named “description”.

- **Special-Characters Standard.** No characters other than letters, numbers, periods, or hyphens are allowed in the name. For example, the following characters are not allowed: slashes (/ or \), ampersands (&), asterisks (*), dollar signs ($), pound signs (#), apostrophes (‘), quotes (“), underscores (_), etc.

- **Spaces-Separator Standard.** Use spaces to separate terms within the name.

- **Lower-Case Standard.** Use lower case letters, except when acronyms that are generally capitalized are included. Thus, the following are acceptable attribute names: “employment status code”, “GL account”.

- **Name-Restriction Standard.** Attribute names must not contain any of the following:
  
  - Words in possessive form (e.g., product’s unit cost)
  - Indefinite or definite articles (e.g., “a”, “an”, “the”)
4. Best Practices for Attribute Names contained in Classes or Data Structures

- **Describe-Business-Meaning Best Practice.** Each attribute should be given a name that represents its business meaning (i.e., a name that has meaning to the user community).

- **Use-of-Adjectives Best Practice.** Most attribute names should contain an adjective(s) to differentiate information about the basic concept, where appropriate (e.g., “current residence state code”).

- **Avoid-Repeating-Class or Data Structure Name Best Practice.** In general, the name of the class or data structure should not appear in the name of the attribute. For example, the class “Person” should not have an attribute named, “person name”; rather the attribute “name” would be sufficient. An exception to this rule includes cases where the attribute name is well-known when it includes the class or data structure name (e.g., the class “Signal” may have an attribute named “signal to noise ratio”, because “signal-to-noise ratio” is the name of a well-known concept).

- **Avoid the Use of the Full Class Name and Avoid the Use of “Value”**. Sometimes it may seem that the best name for an attribute would be its full class or data structure name or the word “value”. An attribute named to match the full class or data structure name would be verbose and duplicative. And naming the attribute “value” has the opposite effect in that this name is not very clear. The preferred approach would be to select a portion of the class or data structure name as the attribute name. Table C-1 shows the preferred approach of selecting a portion of the class name as the attribute name.
Table C-1 Examples Showing Preferred Way to Name Attributes

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Candidate Attribute Name</th>
<th>Preferred Attribute Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone Number</td>
<td>phone number</td>
<td>number</td>
</tr>
<tr>
<td>Report Title</td>
<td>report title</td>
<td>title</td>
</tr>
<tr>
<td>Person Skill</td>
<td>person skill</td>
<td>skill</td>
</tr>
</tbody>
</table>

- **Common-Attribute-Name Best Practice.** The following attributes tend to be needed for quite a few classes or data structures within data models. When incorporating these concepts, the attributes should be named, defined, and sourced, if appropriate, as noted in Table C-2:

Table C-2 Common Attribute Names and Suggested Definitions/Sources

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Suggested Definition</th>
<th>Source</th>
<th>Example Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin date</td>
<td>The date that &lt;whatever you are describing&gt; started.</td>
<td>COMMON</td>
<td>The date that the lawyer’s assignment to the case started.</td>
</tr>
<tr>
<td>begin date time</td>
<td>The date and time that &lt;whatever you are describing&gt; started.</td>
<td>COMMON</td>
<td>The date and time that the lawyer’s assignment to the case started.</td>
</tr>
<tr>
<td>end date</td>
<td>The date that &lt;whatever you are describing&gt; terminated/stopped/completed.</td>
<td>COMMON</td>
<td>The date that the lawyer’s assignment to the case terminated.</td>
</tr>
<tr>
<td>end date time</td>
<td>The date and time that &lt;whatever you are describing&gt; terminated/stopped/completed.</td>
<td>COMMON</td>
<td>The date and time that the lawyer’s assignment to the case terminated.</td>
</tr>
<tr>
<td>is &lt;adjective&gt;</td>
<td>A two-state variable &quot;true or false&quot; to indicate whether &lt;whatever you are describing&gt; is &lt;adjective&gt;.</td>
<td>COMMON</td>
<td>A two-state variable (true or false) to indicate whether a report is distributed.</td>
</tr>
<tr>
<td>name</td>
<td>A word or words by which a/an &lt;class or concept name&gt; is known.</td>
<td>COMMON</td>
<td>A word or words by which a language is known.</td>
</tr>
<tr>
<td>note</td>
<td>A comment to provide background, contextual, or explanatory information.</td>
<td>COMMON</td>
<td>A comment to provide background, contextual, or explanatory information.</td>
</tr>
<tr>
<td>title</td>
<td>An identifying or descriptive name given to a(n) &lt;class name&gt;.</td>
<td>COMMON</td>
<td>An identifying or descriptive name given to a report.</td>
</tr>
</tbody>
</table>

PD Guidance 147

To assist the data modelers, PowerDesigner extensions allow for easy insertion of definitions for common attribute names. To insert a common definition for an attribute, go to the Attribute properties worksheet. General tab, and click on the Common button at the right of the Definition section. A Common Defs window will open. Choose one of the eight buttons corresponding to the rows in Table C-2. An appropriate definition and source will be placed in the Definition and Source sections of the Attribute properties worksheet. You may edit the suggested definition if needed.
• **One-Idea-or-Concept Best Practice.** The attribute name should not represent more than one idea or concept.

• **Representational Class Word Best Practice.** In general, representational class words will not be used. The following words, however, should be included as part of an attribute name, where applicable.
  
<table>
<thead>
<tr>
<th>Word</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Include when the attribute represents a date (e.g., “begin date”).</td>
</tr>
<tr>
<td>Date time</td>
<td>Include when the attribute represents a date time (e.g., “begin date time”). For date or date time attributes other than the common ones in Table C-2, if the datetime attribute name includes a noun, name it as &quot;&lt;noun&gt; datetime&quot; (e.g., creation datetime). If the datetime attribute name includes a verb, name it as &quot;datetime &lt;verb&gt;&quot; (e.g., datetime created).</td>
</tr>
</tbody>
</table>

• **“Type” Best Practice.** In general, it is worth considering whether the name “type” is sufficient for an attribute, especially for a partitioning attribute. It is often much more revealing to indicate a reason for the subdivision into different types. Consider, for example, “usage type” when the division is based on functional usage; consider “composition type” when the division is based on the material from which an instance of the class is fabricated.
APPENDIX D. ELDM ACRONYM GUIDELINES

The purpose of this appendix is to provide principles for when and how to incorporate acronyms within the names of concepts in an ELDM.

An acronym is a word formed by:

- Using the first letter of each word (or each key word) in a name or series of words. For example, the acronym “SSN” for “Social Security Number”; or the acronym “POC” for “Point of Contact” or

- Combining the initial letters or parts of a series of words. For example, the acronym “radar” for “radio detecting and ranging”.

Acronyms are allowed within the name of a model, business area, diagram, class, attribute, primitive data type, domain, enumerated list, and data structure within an ELDM. The centralized ELDM management team will maintain a list of approved acronyms. This list will be maintained outside the ELDM. The following principles will be followed deciding whether and how to incorporate acronyms.

- **Well-Known Guideline**: In general, only acronyms that are well-known by all members of the business should be used. Thus, the acronym should be known and understood by members of the business both inside and outside the business area being modeled.

- **Use Common Case Guideline**: For clarity in communicating, acronyms are to be written in the case in which they are most commonly communicated. In most cases, this will be all upper case (e.g., SSN; not ssn). There are, however, exceptions to this, where a combination of upper and lower case is the most common form (e.g., IPv4, IPv6, DoD, etc.).

  **Note**: When an ELDM is transformed into a PDM, these acronyms will need to be converted to upper camel case, upper case, lower case, or whatever case is needed for the physical implementation.

- **No Period Standard**: Periods should not be used with acronyms (i.e., use “US”; not “U.S.”).

- **Re-Use Guideline**: An acronym may be re-used within an ELDM as long as the full name of the concept satisfies uniqueness rules (e.g., the full name of the class must be unique within the ELDM). For example, it would be acceptable to use “COI” within one class name to mean “Conflict of Interest” and for another class name to use this same acronym to mean “Community of Interest”.

- **Special Guideline for Multiple, Related Classes Using the Same Acronym**: If one class name consists of a word or words that represent a common acronym and additional, related class names consist of the same word or words that represent the same common acronym plus other word(s), then the first class should be fully spelled out and the additional classes may be
named with the acronym plus the additional word(s). For example, if one class represents a “Special Order”, if “SO” is a common acronym, and if a another, related class represents a special order line item, then the first class should be fully spelled out as “Special Order” and the other class may be named “SO Line Item”.

• In order to remove any ambiguity that may be present as a result of using an acronym in the name of the concept, the fully spelled-out form of the name shall be placed in the “Note” property.
APPENDIX E. ELDM DEFINITION GUIDELINES

The purpose of this appendix is to provide guidelines for developing high quality definitions for all concepts within an ELDM, especially classes, data types, and attributes. The essence of these guidelines is that all concept definitions are to be clear and understandable to all members of the business.

Every concept definition should conform to the following set of rules.

- The definition should be a statement of what the concept is. \(^{vii}\)

- The definition should include only the set of characteristics of the concept which are essential (or indispensable) for the proper understanding of the concept. Types of characteristics include composition, color, function, use, origin, shape, location, and movement. For example, Table E-1 shows a number of characteristics of a “Wooden Pencil”. Some of the characteristics are essential and others are not essential. “Absence of an essential characteristic fundamentally changes the concept”. \(^{viii}\) The italicized example following the table presents a good definition based on the table.

Table E-1 Characteristics of a “Wooden Pencil”

<table>
<thead>
<tr>
<th>Type of characteristic</th>
<th>Characteristic</th>
<th>Essential or Non-essential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composition</td>
<td>Graphite core</td>
<td>Essential</td>
</tr>
<tr>
<td>Composition</td>
<td>Graphite is encased in wood</td>
<td>Essential</td>
</tr>
<tr>
<td>Color</td>
<td>Casing may be colored</td>
<td>Non-essential</td>
</tr>
<tr>
<td>Composition</td>
<td>One end may have an eraser</td>
<td>Non-essential</td>
</tr>
<tr>
<td>Shape</td>
<td>One end may be sharpened to a point</td>
<td>Non-essential</td>
</tr>
<tr>
<td>Usage</td>
<td>Must be sharpened for usage</td>
<td>Essential</td>
</tr>
<tr>
<td>Medium</td>
<td>Graphite is the writing medium</td>
<td>Essential</td>
</tr>
</tbody>
</table>

- Good definition example:

- Wooden Pencil: a utensil consisting of a graphite core writing medium encased in wood, usable for writing when sharpened at one end to expose a graphite point.

- The definition of a concept should only contain information that makes the concept unique. Any additional information deemed necessary should be included in the “note” property. \(^{ix}\)

- The definition of a concept should only describe that one concept. It should not include a definition for any other concept or term. \(^{x}\)

- The definition should build upon, support, and be consistent with the name. Once a concept has been named and defined, an assessment should be made to determine whether the meaning suggested by the name matches the definition. Contradictions between the implied meaning from the name and the definition lead to usage errors.
The substitution principle should be used to test the validity of a definition. A definition is valid if it can replace a designation in a text without loss of or change in meaning. For example, a business rule may state that “A customer may place an order.” A test for the validity of the definition for the class Order should be whether the definition can replace the word “order” in the statement above. For example, if the class Order is defined as “A request for one or more Products”, the substitution principle would cause one to test whether it makes sense to state, “A customer may place a <request for one or more Products>.” In this case, the definition passes the substitution principle test.

The definition should be as concise as possible, yet as complex as necessary to be clear. “Long rambling prose generally adds little value, and can introduce ideas or concepts not directly related” to the concept, “leading to ambiguity and the potential for misinterpretation.” Stick to the point of the definition.

The definition should be precise and unambiguous. The exact meaning of the definition should be clear enough to allow only one possible interpretation.

The definition should define a single instance of the concept. For example, the class Person might be defined as “A human being.”

The definition should be unique (i.e., there should not be more than one concept in the data model that has the same definition).

The definition of a concept must be true of every instance of the concept.

The definition should be validated against real world instances of the concept.

If the name of a concept includes an acronym the fully spelled-out form of the name shall be placed in the “Note” property (i.e., the definition for a concept shall not begin with the name of the concept).

The following should be avoided in a definition:

The definition for a concept should not be too broad (i.e., allowing instances of objects to be included that are not appropriate). For example, if Mechanical Pencil were defined as “A writing instrument that is composed of a barrel and a refill” it would be too broad as it does not specify the type of refills. Thus, the definition would allow ball-point pens, roller-ball pens, and felt-tip pens to be included, which should not be allowed. To resolve this problem, the definition should be revised to “A writing instrument composed of a barrel, a lead refill, and a lead-advance mechanism.”

The definition for a concept should not be too narrow (i.e., excluding instances of objects that should be allowed). For example, if Mechanical Pencil were defined as “A writing instrument composed of a barrel, a lead refill, and push-button advance mechanism” it would be too
narrow as the inclusion of the “push-button advance mechanism” characteristic excludes those mechanical pencils that use other types of advance mechanisms. To resolve this problem, the definition should be revised to “A writing instrument composed of a barrel, a lead refill, and a lead-advance mechanism.”

- The definition of a concept should not be circular in which the term being defined is part of its own definition. For example, the following definition for Cordless Phone is circular: “A telephone that has no cord.”

- The definitions within a set of concepts should not be circular (i.e., where two or more concepts are defined by means of each other). For example, the following two definitions are circular:
  
  o “Virgin Forest” – “A forest constituted of a natural tree stand.”
  
  o “Natural Tree Stand” – “A stand of trees grown in a virgin forest.”

**Note:** One way to resolve this problem would be to change the definition for “Natural Tree Stand” to remove its circularity (e.g., “A stand of trees grown without interference by man”). In this case, the definition for “Virgin Forest” would not need to be changed.

- The definition should not simply restate or rephrase the name of the concept.

- The definition, in general, should avoid using the name of the concept being defined, unless the name is a common, well-understood word or term. For example, the class Medical Treatment should not be defined as “A treatment for an illness or condition”; rather, it should be defined as “Care given to a person that is intended to relieve an illness or condition.” However, it would likely be acceptable to define the attribute password as “The password of the User Account”, since the term “password” is well-known and readily understood.

- The definition for a concept generally should not even begin with the name of the concept. For example, the definition of the class Person should not be “A person is a human being.” Rather, the definition merely needs to be “A human being.”

- The definition should not use technical jargon or include acronyms or abbreviations that are not explained. If these things must be used, they should be defined or explained in a note.

- The definition should state what the concept is, not what it is not. For example, if the class Deciduous Tree were defined as “A tree other than an evergreen tree”, it would be deficient as it states what it is not. A corrected definition might be “A tree that loses its foliage seasonally.” An exception to this guideline would be when the absence or non-existence of a characteristic is essential to the understanding of the concept. In this case, a negative definition may be required. For example, “Nonconformity” may be defined as “Non-fulfillment of a specified requirement.”

- The definition of a concept should not include a definition of a subordinate concept.
• The definition of a class should not list the attributes contained in the class.

• The definition of an attribute should not simply restate metadata characteristics about the attribute, such as its length, data type, or permitted values.

• The definition should not include nonessential information. Thus, for example, all of the following should be placed in the “note” property:
  
  o Examples – Even though examples with broad applicability can be an excellent way to supplement a definition, they do not provide enough of the business meaning necessary for a definition and consequently belong in the “note” property. In addition to placing examples in the “note” property, they may be recorded as a note directly on an entity relationship diagram (or class diagram), if appropriate.

  o Nonessential characteristics of the concept.

  o A definition or explanation for any of the characteristics that are used to define the concept.

  o Comments about the concept that are not essential to the definition.

• The definition should not contain other names used by the business for the concept. Other names should be placed in the “Alias” property.

PD Guidance 148  Tokenization

A number of tokens have been defined to streamline and simplify the entry and maintenance of definitions. Each token can be used to reference an object related in some way to the object the modeler is defining. As explained in APPENDIX B, definitions are entered or modified in a special “Edit Definition” window. Tokens are entered in this special window and interpreted in the actual Definition field.

The primary advantage of these tokens is that PowerDesigner maintains the correct reference to the other object even if the modelers change the name of the object in the course of model development. This saves the modelers from needing to update potentially many definitions simply because the name of one object has changed.

After tokens are entered, the tokens as entered will usually be displayed in the “Edit Definition” window, while the current interpretation of the token will appear in the Definition field. The only exception to this is the [[&Name&]] token. The exception is explained below.

The following are the available tokens:

- [[[$me]]]
  The name of the object you defined.
The name of the supertype of the object you defined.

The name of the subtype of the object you defined. Note that if there are multiple subtypes (which is normally the case), one of them will be selected, by a method which is out of your control. So this token should rarely, if ever, be used.

The name of the parent of the object you defined. (If defining an attribute, it will be the parent class. If defining a class, it will be the parent package.)

For Associations, the name of the object at the beginning of the association, meaning the object from which the association was originally drawn. Note that this might not correspond to the direction of the arrowhead of the association if the navigability of the association was changed after its creation. ([$A$] resolves to the class in the position of “Class A” when you open the Association Properties Worksheet.)

For Associations, the name of the object at the end of the association, meaning the object to which the association was originally drawn. Note that this might not correspond to the direction of the arrowhead of the association if the navigability of the association was changed after its creation. ($B$ resolves to the class in the position of “Class B” when you open the Association Properties Worksheet.)

NOTE! Name is case sensitive.

"Name" is the name of a class referenced by the definition. The name may be either the "Name" or the "Code", the script will locate an object with either. The [[&Name&]] is replaced in the editable definition. The replacement format is [[(Name)<object uuid>]]. The result in the editable definition is ugly, e.g.,[[(Person){49073592-C56A-4B30-A5ED-8D52758AA770}]] and makes the editable definition more difficult to read. The payoff is that if "Person" is changed to "User", the definition will be changed as well.

Examples of token usage appear in PD Guidance for some of the following Definition Guidelines.
The following definition guidelines apply to high-level ELDM concepts:

**Business Area Definition Guideline**

- Since a business area is normally focused on one independent class, and named appropriately, its definition can simply state “The portion of the Enterprise Data Model focused on the **name of primary focus class** class.” For example, the definition of the **Party** business area would be “The portion of the Enterprise Data Model focused on the **Party** class.”

**Class Diagram Definition Guidelines**

- The main diagram of a business area should be simply defined as “The main diagram of the **name of business area** business area.” For example, the definition of the main diagram of the **Party** business area would be “The main diagram of the **Party** business area.”

- Other diagrams of a business area should be defined as “A diagram of the **name of business area** business area focused on the **name of primary focus class of diagram** class.”

The following definition guidelines apply to specific stereotypes of classes:

**Kind Definition Guideline**

- The definition for a **<<kind>>** class can usually be derived from or taken directly from an open source, such as the Concise Oxford English Dictionary. For example, the **Person** class in Figure E-1 could be defined as “A human being.”

```
<<kind>>
Person

name : string
height : float
weight : float
gender : genderEnum
```

*Figure E-1: Defining a **<<kind>>** Class*

**Role Definition Guideline**

- The definition for a **<<role>>** class should reference its parent **<<kind>>** or **<<role>>** class, as appropriate. A suggested definition template is: “A/an **<parent kind or role class>** who/that...<describe what caused the role to exist or how the business uses the role>.” For example, the **Doctor** class in Figure E-2 could be defined as “A **Person** who has received a license to practice in the healing arts.”
Using tokens, the above example would be entered as “A \[[$sup]\] who has received a license to practice in the healing arts.”

**Category Definition Guideline**

- The definition for a <<category>> class should reference its <<kind>> or <<category>> child classes. For example, the Party class in Figure E-3, could be defined as “A Person or Organization.”

Using tokens, the above example would be entered as “A [[&Person&]] or [[&Organization&]].”
Role Category Definition Guidelines

- The definition for a <<role category>> class should reference the <<kind>> superclasses of the “child” classes if all the <<kind>> superclasses are known and if the number of “child” <<role>> or other <<role category>> classes is a small number (e.g., seven or less). A suggested definition template is: "A/an kind superclass of Child 1, kind superclass of Child 2 ... or kind superclass of Child n who/that ...<describe what caused the role category to exist or how the business uses the role category>." For example, the Resource class in Figure E-4, could be defined as "A Person, Organization or Device that is able to perform work."

- Alternatively, the definition for a <<role category>> class could reference the <<category>> superclass of the <<kind>> superclasses of the “child” classes if all the <<kind>> superclasses are known and a <<category>> superclass for them has been defined. A suggested definition template is: "A/an category superclass of all children who/that ...<describe what caused the role category to exist or how the business uses the role category>." For example, the Customer class in Figure E-5, could be defined as "A Party that acts as a recipient of goods or services in a transaction."
In the event that all the <<kind>> superclasses are not known, or their number can reasonably be expected to increase, the definition for a <<role category>> class should be written more generically. A suggested definition template is: "Any class who/that ...<describe what caused the role category to exist or how the business uses the role category>.

**Dependent Definition Guideline**

- The definition for a <<dependent>> class should reference the class upon which the <<dependent>> class depends for its existence. For example, the Person Skill class in Figure E-6 could be defined as “An ability possessed by a Person.”
The definition of an <<associative>> class should reference the classes upon which the <<associative>> class depends for its existence. A suggested definition template is: "A mapping representing the fact that a specific <class 1 from which identity is passed to the associative class> <verb phrase> a specific <class 2 from which identity is passed to the associative class>.

For example, the Financial Institution Account Mapping class in Figure E-7, could be defined as “A mapping representing the fact that a specific Account is held with a specific Financial Institution."

The definition of an <<event>> class should be described as a period or point in time during which an activity happens, and should reference the most important associated classes, such as participants in, or location of, the happening. For example, the Employment class in Figure E-8 could be defined as “The period of time during which an Employee works for an Employer under a formal or informal Employment Contract.”
The following additional definition guidelines apply to classes:

- The definition of a subclass class (regardless of its stereotype) should generally reference its superclass class. For example, the definition of **Helicopter** as a subclass of **Aircraft** might be “A type of **Aircraft** that is propelled by an overhead rotor and a directional rotor in the tail.”

- When defining classes within a network of related classes (e.g., for a superclass and its subclasses), the definition for each class should highlight or concentrate on the delimiting characteristics (i.e., those characteristics that are not only essential for the proper understanding of the class, but also distinguishes the class from others in the network, or from the superclass and other subclasses in this case). For example, when defining a **Mechanical Pencil** as a subclass of a **Pencil**, the definition might be “A type of **Pencil** with a permanent outer casing and with a graphite core that advances for usage.”

The following definition guidelines apply to attributes:

**Attribute Definition Guidelines**

- See Table C-2 for suggested definitions for the “common” attributes: begin date, begin date time, end date, end date time, (boolean) is <adjective>, name, note, and title.

- Partitioning attributes should be defined as “A subdivision of the **name of class being partitioned** class based on (the reason or characteristic of the partitioning).” For example, if the **Student** class had a partitioning attribute to separate in-state and out-of-state students, the attribute might be defined as “A subdivision of the **Student** class based on state residency status.”
Using tokens, the above example would be entered as “A subdivision of the [[$pc]] class based on state residency status.”

- The definitions of other attributes should convey the use of that attribute in describing, limiting or defining the class in which it is located and, as such, should normally contain “name of class” within the definition.

The following definition guidelines apply to data types:

- Since a data type is a means of specifying values which may be used and reused for multiple attributes, the definition of a date type should not generally reference any attribute which uses it.

**Enum Definition Guideline**

- A suggested definition pattern for an enumerated list (enum) is: "A list of allowable values for attributes which require a defined set of values for (the general usage expected) using (the form or source of the values).” Often the source of the values will be a standard, such as an ISO standard. For example, the **Country Code Enum** data type in Figure E-9 could be defined as "A list of allowable values for attributes which represent or are described by country names, using a set of trigraphic representations."

```
<<kind>>
Country
country code : countryCodeEnum
```

![Figure E-9: Defining an <<enum>> Class](image)

**Data Structure Definition Guideline**

- A good pattern for a data structure definition is “A data type used for specifying (a multidimensional concept) using (two or more descriptions of attributes).” For example, the **Color Struct** data structure in Figure E-10 could be defined as “A data structure for specifying a color of a class using floating point hue, saturation, and intensity values.” As another example, a **Time Interval Structure** could be defined as “A data structure for specifying a continuous period of time using beginning date-time and end date-time points.”
Figure E-10: Defining a <<struct>> Data Type

**Domain Definition Guideline**

- A recommended pattern for a domain definition is “A data type which restricts a(n) (associated primitive data type) to (the restriction).” For example, a good definition for the Positive Float Domain would be “A data type which restricts a float to values greater than zero.” As another example, a good definition for the US Zip Code Domain would be “A data type which restricts a string to values conforming to the US Postal Service regular expression governing zip codes.”

**Union Definition Guideline**

- A recommended pattern for a union definition is “A list of data types [optionally provide the list here] which together provide alternative mechanisms that can be used for (the general usage expected).” For example, the **Spatial Position Struct Union** might be defined as “A list of data types (specifically the coordinate reference systems Geodetic 2D Struct, Geodetic 3D Struct, and Cartesian Struct) which together provide alternative mechanisms that can be used for describing a location in space.”
APPENDIX F.  ELDM DEFINITION SOURCING GUIDELINES

The purpose of this appendix is to specify guidelines for how to properly “source” a definition in an ELDM.

All defined concepts in an ELDM must have a “definition primary source” property and may also have a “definition source detail” property.

This appendix consists of general guidelines for sourcing definitions, describes how a source citation should be formatted, and includes sample citations.

General Guidelines for Sourcing Definitions

The following are general guidelines for sourcing definitions:

- As discussed in APPENDIX B (ELDM Common Properties) there are two properties established for the purpose of capturing definition source information:

  1. The “definition primary source” property specifies a citation for the main reference from which the knowledge to develop the definition was obtained or derived.

     Notes: For an ELDM, a list of commonly used sources should be developed from which a modeler may select one to apply as the primary source for a definition. A single source should only appear once in this list.

     In general, the “definition primary source” property should not include a web address (as this is a detail that is more appropriate for the “definition source detail” property and since some sources are available from multiple websites, making this web address a non-dependency fact). However, if a source is only identifiable with a web address, the “definition primary source” property may state the web address from which the source was retrieved (e.g., “Retrieved from <web address>”).

  2. The “definition source detail” property specifies additional information about the source(s) used to define a concept (i.e., beyond just the primary source citation). This property provides a way for those who record definitions to communicate in a more detailed way exactly where the information was obtained to develop the definition. Examples of the types of additional information that could be provided include:

     o The fact that the definition was directly extracted from the primary source or was derived from the primary source (e.g., "Derived from <short name of source>").

     o Specific details, as appropriate, regarding the section(s) from within the source that were used to extract or derive the definition. Examples include:

       ➢ The term(s) referenced from the source

       ➢ The part(s) of speech of the term(s)
The definition number(s) of the term(s).

Since the potential sources are extremely diverse, the precise format of the definition source detail is not prescribed. The following is a good example of source detail where the definition was derived from a dictionary:

"Derived from definitions for <term 1>, definition 3 and <term 2>, definition 5."

- Additional source(s) used (i.e., that, like the primary source, also accurately reflect the intended meaning of the concept).

**Note:** All source citations must be formatted as described below in this appendix.

- The web address and date retrieved if the publication was retrieved from the internet. For example, “Retrieved on October 14, 2008 from http://www.psychologytoday.org/journals/webref.html.”

**Note:** A hard return should precede detail about the second and subsequent entries in the “definition source detail” property. For example, a “definition source detail” property may state that the definition was derived from the primary source and a new line should be started to add that an additional source was used.

- A primary source must be cited for each definition (in the “definition primary source” property).

- Additional sources may be cited in the “definition source detail” property.

- Any source that is cited in either the “definition primary source” or “definition source detail” properties must be one that accurately reflects the intended meaning of the concept (i.e., the intended meaning of an instance of the concept). For example, there could be multiple definitions for the word “Resource” in a variety of sources. The source(s) that are cited for the class **Resource**, however, must define Resource in the way intended by the model (which, of course, should be the way it needs to be defined by the business).

- The proper formatting for a source citation (in either property) must conform to the guidelines shown below.

- A source can be quoted verbatim or can be wordsmithed, as needed, in order to most accurately define the concept.

- If the wording from a source is wordsmithed, the "definition source detail" property should indicate this fact, using the words, “Derived from <source>” (e.g., “Derived from Concise Oxford English Dictionary”).

- Where possible, citations should be unclassified.
A security portion marking is not needed for a "definition primary source" or "definition source detail" properties, since the definition already has a “definition security marking” property, which should be the highest security marking for the “definition”, “definition primary source”, and “definition source detail” properties.

For definitions obtained from a web site, cite the publication found at the web site (e.g., “RFC 1939”), the name of the information obtained from the web site (e.g., “Organization X Common Definitions”), or the name of the web site (e.g., “Organization X Homepage”). Do not merely provide the web address.

Figure F-1 below depicts a decision tree that should be used to select the “primary” source if multiple sources provide a definition that accurately reflects the intended meaning of the concept. An explanation of the steps involved in this decision tree follows:

Step A: Is the concept to be defined one of the common attributes shown in Table C-2 (Common Attribute Names and Suggested Definitions/Sources) of APPENDIX C (ELDM Naming Guidelines)?

1. If so, that definition should be used and the primary source shown as “COMMON” as indicated in the table.

Step B: Was the concept created to satisfy a data modeling convention required by this DMG that would not be readily recognized by members of the business?

Note: The following items are examples of concepts where this is likely the case:

- An attribute for which an enumerated list has been created, where the enumerated list is not one maintained by the business community (e.g., assessment status)
- A partitioning attribute added to differentiate among class subclasses, where the names of the subclasses are not readily recognized by members of the business (e.g., “request type” to differentiate between an “Information Request” and a “Service Request”)
- A class created to represent a set of highly cohesive attributes, where the set would not be readily recognized by members of the business (e.g., “USA Caveat”)
- An associative class that would not be readily recognized by members of the business (e.g., Person Language Mapping).

2. If so, cite "INTERNAL" as the complete source citation in the “definition primary source” property.

Note: When “INTERNAL” is cited as the primary source, it indicates that the centralized ELDM team developed the definition without using an external source.
Data Modeling Guide (DMG) For An Enterprise Logical Data Model, V2.3; 15 March 2011

Step C: Is the item a common concept (i.e., one that is used outside the cryptologic and classified domains)?

   If so, proceed to step D

   If not, proceed to step F.

Step D: Is the item a non-technical concept?

   3. If so, cite the Concise Oxford English Dictionary.

   If not, proceed to step E.

Step E: Is the item a technical concept whose definition can be found in one of the preferred technical sources found on a list maintained by the enterprise organization?

   4. If so, obtain the definition from and cite one of the preferred technical sources

   5. Otherwise, cite another technical source.

Step F: Can the definition for the item be found in an “internal” unclassified source (e.g., an unclassified source only available within the enterprise organization)?

   6. If so, cite this source

   7. Otherwise, cite an internal, classified source.

If this decision tree fails to yield an appropriate source, an organization may be cited. However, it is expected that an organization would only be cited if the concept represents a proprietary product produced by that organization and no other source documentation exists.

A person should not be used as a source unless there is no other acceptable source.
Figure F-1: Decision Tree to Select “Primary” Source
How a Source Citation Should be Formatted

As stated above, a source citation will be placed in the “definition primary source” property and may also be placed in the “definition source detail” property. In both cases, the source citation should be recorded using the following guidelines:

A source citation consists of the following elements, where applicable, in the order shown here:

1. Author(s):
   - Invert all authors’ names (i.e., listing the surname followed by the author’s initials, with a period after each initial, such as “Baker, F. M.”)
   - With two or more authors, use an ampersand (&) before the last author
   - Use commas to separate authors, to separate surnames and initials, and to separate initials and suffixes (e.g., “Smith, W. R. Jr.” and “Johnson, B., III”)
   - List only up to six author names
   - When there are seven or more authors, abbreviate the seventh and subsequent authors as “et al”
   - Finish the author element with a period.

2. Short name of the source:
   - Give the abbreviated form of the name of the source, if appropriate. If an abbreviated form is not appropriate (e.g., Hargrave’s Communications Dictionary), give the long name of the source in this first position
   - Omit the words “The,” “A” or “An” at the beginning of the name
   - Include the edition number (i.e., 4, 11, etc.) in the name if one exists, but abbreviate edition as “ed”
   - Follow the short name with a space, hyphen, and space (e.g., “-”)
   - This is a required element.

3. Long name of the source:
   - List the long name of the source in Title Case
• The long name should generally not include abbreviations (unless the published name includes an abbreviation)

• Finish the long name element with a period

• This is a required element, unless the long name matches the short name


4. Where published and publisher name:

• List where published and then list the publisher’s name. For example, “New York: Oxford University Press.”

• Where published:

  o Give the city and, if the city is not well known for publishing or could be confused with another location, the state or province (and/or country) where the publisher is located. Use a colon after the location. For example, “New York:” or “St. Louis, MO.”

  o Use the official two-letter U.S. Postal Service abbreviations for U.S. states and territories, if needed.

  o If two or more publisher locations are given, give the location listed first in the book or, if specified, the location of the publisher’s home office.

• Publisher name:

  o Give the name of the publisher in as brief a form as in intelligible. Write out the names and associations, corporations, and university presses, but omit superfluous terms, such as “Publishers, Co.”, or “Inc.”, which are not required to identify the publisher. Retain the words “Books” and “Press”. For example, “IEEE Press.”

  o If the publisher is a university and the name of the state or province is included in the name of the university, do not repeat the state or province in the publisher location.

• Finish the element with a period.

5. When published:

• For non-periodicals, give the year the work was copyrighted. For example, “2005.”

• For periodicals, give the month, day (if available) and year on the publication. For example, “February 16, 2004.” or “June, 2007.”

F-7
• Finish the element with a period.

6. Web address from which retrieved for electronic sources:

• In general, the web address from which an electronic source was retrieved should be placed in the “definition source detail” property. However, if a source is only identifiable by the web address, this information should be placed in the “definition primary source” property.

• If the retrieval information will be placed in the “definition primary source” property, the citation should state, “Retrieved from <web address>.” If it will be placed in the “definition source detail” property, the citation should state, “Retrieved on <date> from <web address>.” The reason for the difference is that the values in the “definition primary source” properties may be reused by other definitions and retrieval information specific to only one definition should not be included.

• If the web address is available on a special network, this should be indicated.

Sample Citations

The following are sample citations that may be found in the “definition primary source” or “definition source detail” properties:


  Note: This is an example where the short name for the source is the same as the long name.


  Note: in the “definition source detail” property, the following may be included: “Retrieved on <date> from http://www.ietf.org/rfc/rfc1939.txt”


  Note: in the “definition source detail” property, the following may be included: “Retrieved on <date> from http://www.itu.int/rec/T-REC-T.30-200509-1/en”


APPENDIX G. ADDITIONAL ELDM GUIDELINES

The purpose of this appendix is to specify guidelines for how to model various patterns of data concepts that occur often in data modeling. Specifically, this appendix addresses the following principles and patterns:

1. “Snapshot in Time” Principle
2. “Perfect Knowledge” Principle
3. “Alternative to XOR Boolean Constraint” Pattern

1. “Snapshot in Time” Principle

Every organization/business must deal with the fact that its data can change over time. For example, an employee has a “salary”, but at some point in the future the “salary” will likely be a new, higher amount. Many organizations must keep track of each value for an attribute as it changes over time (along with the date that the change took place). Keeping track of each value for an attribute as it changes over time is referred to as keeping track of “historical” data. Doing this facilitates the successful completion of audits, data validations, and queries, which often require knowledge of historical data values.

For an ELDM, all data models should reflect a “snapshot in time” perspective. In other words, a data model should only reflect the values that can apply at a single point in time. Thus, an Employee class would be expected to only have one salary attribute in an ELDM. It would still be valid, however, for a person to have multiple phone numbers, since even at a single point in time a person could have a work phone number, a home phone number, a mobile phone number, etc.

When an ELDM is used to generate a PDM, there will be at least three possible choices for how to deal with the historical aspects of data:

1. Ignore the fact that the data can change over time and only capture the current or latest value for an attribute.
2. Only record the “first heard date” and “last heard date” for an attribute or set of attributes. For example, if the business wanted to keep track of a person’s name, one could record the name along with the date of the first time that this name was obtained for the person; and then each subsequent time that the same name was obtained for this person, the “last heard date” would be updated to reflect this latest date heard.
3. Modify the physical data model to allow every change in value to be recorded (along with the date of each change in value). For example, if the business needed to keep track of an employee’s salary as it changes over time, the following changes would need to be made to the physical data model:
• Add an **Employee Salary** class

• Add a dependency association between the **Employee** and **Employee Salary** classes

• Move the **salary** attribute from **Employee** to **Employee Salary**

• Add an **effective date** attribute to the **Employee Salary** class.

2. **“Perfect Knowledge” Principle**

Similarly, throughout an ELDM, the model should be developed to reflect the data that could exist if the business had “perfect knowledge” (i.e., if the business were able to obtain all of the facts that are valid that the business cared about). Following this principle will primarily affect:

- An attribute’s “is mandatory” property and

- An association’s cardinality (specifically the “optionality” fact).

For example, if the **Person** class had an attribute named **weight**, this attribute should be marked as “mandatory”, because with perfect knowledge about any person, this attribute would be populated. This is despite the fact that the business might not obtain someone’s weight in a particular situation. Likewise, if the business needed to perform a biographical study of a group of people and was interested in knowing the roles that a person had, such as “Doctor”, and if the business knew that being a doctor required a certification from a licensing organization, then the **Doctor role** should show a mandatory association to the licensing event that must have taken place. The business knows that this event must have taken place; it is just a matter of getting the needed information. Further, not having the information just points out areas where the additional information needs to be obtained.

The reason the “perfect knowledge” principle is being applied to an ELDM is to more clearly record when an attribute or association cardinality may be optional. Optionality should not be allowed in the model if it is due to the value not being obtained even when it is known that the value must exist (in reality) or if it is due to data corruption that might take place during the process of ingesting the data. Optionality recorded in an ELDM should always and only mean that the business fact (i.e., attribute value or association) does not have to exist in order to reflect reality. A physical model’s optionality facts should reflect the constraints that apply to the implementation environment and so may be different from the ELDM (e.g., there will likely be more attributes and relationships made optional in a PDM).

Nevertheless, there will still be times when it is appropriate in an ELDM to mark an attribute as optional or to have an optional association. Optionality is allowed when, even with perfect knowledge, there may not be a value for the attribute or the association may not exist. For example, as described in the definition of the “is mandatory” property for attributes, if a class has an **end date** attribute, if with
perfect knowledge the “end date” may not be populated, and if it is not appropriate to subclass the class, the **end date** attribute should be marked as “optional”.

**Note:** It is understood that in reality, in certain contexts, especially in those areas where research types of activities take place, the information is often not obtained. In other contexts, however, the information might always be obtained. Thus, it is at the point of implementation that the mandatory/optional notations can be changed to reflect the business’s needs for the specific implementation. For example, in the research area, almost all mandatory cardinalities would need to be made optional. In an employee context, a number of the associations and attributes might not even apply; or if they did, they might be left as mandatory.

### 3. “Alternative to XOR Boolean Constraint” Pattern

Sometimes data modelers are faced with a modeling challenge where a class has two optional associations and any instance of the class must have only one of the two optional associations (but not both associations). This is referred to as an “XOR” Boolean constraint on the two associations.

Rather than specifying a Boolean constraint on the two associations, the data modeler should:

- Create a superclass class for the two XOR related classes and
- Draw a mandatory association to the superclass class.

For example, if the data modeler found that the business rules for a Report were such that each Report was created either by a person or an organization, the data modeler might be tempted to first create the model shown in Figure G-1.

![Figure G- 1: Apparent XOR Boolean Constraint Scenario](image-url)
Rather than place an XOR constraint on these two associations, the data modeler should create a **Party** superclass class (with **Person** and **Organization** subclasses) and draw a mandatory association from **Report** to one and only one **Party**, as shown in Figure G-2.

![Figure G-2: Approach to Avoid Using Boolean Constraints](image)

**4. “Broad Application of Metadata” Pattern**

At times, it is useful to attach “meta properties” to a set of concepts in a model. For example, suppose a business, perhaps a scientific endeavor, maintains information about various entities in which it is interested. It may observe those entities at different times and under different controlled or uncontrolled circumstances. So, in addition to tracking the direct properties of those entities, the business may want to maintain information about the various observations of those properties. The “Broad Application of Metadata” Pattern has been created as an extension of the modeling methodology for such situations.

The following steps are required for the metadata pattern:

1. Model the “meta” properties you wish to associate to the base entities of the model. This modeling may be as simple as defining a single class with a small number of attributes, or far more complex by including numerous classes, data types and relationships. Choose one class as the representative core Metadata Class. For example, Figure G-3 shows the **Observed Entity** class and its associated enumerations. The **Observed Entity** class is a potential Metadata Class.
2. Create a Business Rule, with the stereotype “metadata” Typically, the name of the Business Rule will be the same as, or similar to, the core Metadata Class.

PD Guidance 152
To set the stereotype of a Business Rule, open the appropriate Business Rule Properties worksheet. In the General tab, Stereotype field pull-down list, select the value “metadata.”

3. Designate the core class as the Metadata Class of the Business Rule.

PD Guidance 153
To designate the Metadata Class, open the appropriate Business Rule Properties worksheet. In the General tab, Metadata Class field pull-down list, select the appropriate class.

4. For each class or property (i.e. each class, attribute or association) to which you wish to associate the meta properties, assign the business rule to that class or property. (Generally, Metadata Business Rules are not assigned to classes or attributes in the logical data model, but rather at the domain or physical data model level.)

PD Guidance 154
To assign a business rule to a class or property, open its concept properties worksheet (i.e. Class Properties, Attribute Properties, or Association Properties worksheet). On the Rules tab, select the (Add Objects) icon. A Selection worksheet will open, containing a list of Business Rules associated with the model. Check the selection box next to the name of the business rule you wish to apply.

Figure G-4 depicts the Metadata Business Rule concept. It shows that there is a one-to-one relationship between Metadata Class and Metadata Business Rule. (One might argue that multiple business rules could attach the same Metadata Class, but in effect those business rules would be identical.) A Metadata Business Rule can describe any number of classes/entities and/or properties. And each class or property can be described by any number of Metadata Business Rules.
Figure G-4: The Metadata Business Rule Concept and Its Properties and Relationships
APPENDIX H. OVERVIEW OF ELDM STEREOTYPES

This appendix has been written in order to:

- Define the concept of a stereotype and explain the reasons stereotypes were added for these ELDM guidelines.
- Provide a consolidated list of all of the stereotypes that apply to concepts within an ELDM;
  - Explain when the stereotypes are to be used in an ELDM
  - Explain how stereotypes should be depicted in an ELDM.

A stereotype is a mechanism to indicate that an instance of a data modeling concept (e.g., a specific class, attribute, etc.) has been changed to represent either a new concept or represent a specialization of its existing concept. The stereotypes that have been added for these ELDM guidelines were incorporated to represent a specialization of an existing concept in order to facilitate either:

- Unique ERD/UML translation rules (i.e., rules used to translate or toggle between notation types – ERD to/from UML - for an ELDM) or
- Unique encoding rules - i.e., rules used to convert:
  - An ELDM to a PDM based on the ELDM
  - A PDM based on the ELDM to XSD or
  - A PDM based on the ELDM to Data Definition Language (DDL).

Table H-1 shows the stereotypes that have been added for an ELDM and explains when each stereotype should be used.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Stereotype</th>
<th>When to be Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>&lt;&lt;kind&gt;&gt;</td>
<td>when class type = Kind</td>
</tr>
<tr>
<td>Class</td>
<td>&lt;&lt;category&gt;&gt;</td>
<td>when class type = Category</td>
</tr>
<tr>
<td>Class</td>
<td>&lt;&lt;role&gt;&gt;</td>
<td>when class type = Role</td>
</tr>
<tr>
<td>Class</td>
<td>&lt;&lt;role category&gt;&gt;</td>
<td>when class type = Role Category</td>
</tr>
<tr>
<td>Class</td>
<td>&lt;&lt;dependent&gt;&gt;</td>
<td>when class type = Dependent</td>
</tr>
<tr>
<td>Class</td>
<td>&lt;&lt;associative&gt;&gt;</td>
<td>when class type = Associative</td>
</tr>
<tr>
<td>Class</td>
<td>&lt;&lt;event&gt;&gt;</td>
<td>when class type = Event</td>
</tr>
<tr>
<td>Relationship</td>
<td>&lt;&lt;dependency&gt;&gt;</td>
<td>when relationship type = Association and relationship is drawn to a class which is existentially dependent on the other class for its existence.</td>
</tr>
</tbody>
</table>
Concept | Stereotype | When to be Used |
---|---|---|
Relationship | <<non-dependency>> | when relationship type = Association and relationship is drawn to a class which is not existentially dependent on the other class for its existence. |
Relationship | <<aggregation>> | when relationship type = Aggregation |
Relationship | <<composition>> | when relationship type = Composition |
Relationship | <<specialization>> | when relationship type = Inheritance and relationship is drawn between classes with identical stereotypes, or between a <<kind>> and a <<role>>. |
Relationship | <<realization>> | when relationship type = Inheritance and relationship is drawn between a <<category>> and a <<kind>> or between a <<role category>> and a <<role>>. |
Data Type | <<enum>> | for all Enumerated Lists |
Data Type | <<struct>> | for all Data Structures |
Data Type | <<domain>> | for all Domains |
Data Type | <<union>> | for all Unions |

All classes will have a stereotype since a stereotype exists for every possible type of class. The stereotype name for a class will always be depicted above the class name and will be shown within double angular brackets (e.g., <<dependent>>).

Although all relationships have a stereotype, the stereotype name will not be shown graphically on diagrams. This is due to the fact that other graphical cues already indicate the stereotype of the relationship (i.e., solid lines for dependency; dashed lines for non-dependency; and an open-headed circle on one end of the relationship for inheritance relationships).

All data types will appear as classes, but will be given a stereotype (i.e., <<enum>>, <<struct>>, <<domain>>, or <<union>>, as appropriate) in order to clearly indicate the essence of the item on diagrams.

Figure H-1 depicts the proper placement for stereotype names for classes and enumerated lists.

![Figure H-1: Proper Placement of Class and Enumerated List Stereotype Names](image-url)
APPENDIX I. Modifications to Guizzardi

Figure I-1 indicates in red how the constructs used in the methodology described in this publication were intentionally changed or omitted from the original Guizzardi constructs.

Figure I-1: Original Guizzardi Constructs intentionally omitted from methodology.
Table I-1 Indicates how some of Guizzardi’s original Class Names were kept, changed or omitted for our methodology.

<table>
<thead>
<tr>
<th>Guizzardi’s Original</th>
<th>Methodology</th>
<th>ERD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Kind</td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Collective</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Phase</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Category</td>
<td></td>
</tr>
<tr>
<td>Role Mixin</td>
<td>Role Category</td>
<td></td>
</tr>
<tr>
<td>Mixin</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Event</td>
<td>Independent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode</th>
<th>Dependent</th>
<th>Dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relator</td>
<td>Associative</td>
<td>Associative</td>
</tr>
</tbody>
</table>

Table I-2 Indicates how Guizzardi’s original Relationship Constructs were altered or combined for our methodology.

<table>
<thead>
<tr>
<th>Guizzardi’s Original</th>
<th>Methodology</th>
<th>ERD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalization</td>
<td>Generalization/Realization</td>
<td>Generalization</td>
</tr>
<tr>
<td>Material</td>
<td>Non-Dependency</td>
<td>Non-Dependency</td>
</tr>
<tr>
<td>Formal</td>
<td>Dependency</td>
<td>Dependency</td>
</tr>
<tr>
<td>Attribution</td>
<td>Category</td>
<td></td>
</tr>
<tr>
<td>Mediation</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Whole/Part/Kind</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Whole/Part/Collective</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Whole/Part/Quantity</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
APPENDIX J. END NOTES

---


ii Ibid.


iv The Data Modeling Handbook, A Best-Practice Approach to Building Quality Data Models

v Ibid.

vi International Organization for Standardization (ISO) 1087-1, Terminology work – Vocabulary – Part 1: Theory and application

vii International Organization for Standardization (ISO) 704, Terminology work – Principles and methods

viii Ibid.

ix Ibid.

x Ibid.

xi Ibid.

xii International Standards Organization/International Electrotechnical Commission 11179

xiii Ibid.

xiv Ibid.

xv Ibid.

xvi International Organization for Standardization (ISO) 704, Terminology work – Principles and methods

xvii Ibid.

xviii Ibid.

xix Ibid.

xx Ibid.

xxi Ibid.