

Systems Modeling Language (SysML) Specification Addendum to SysML v. 0.9

Profiles and Model Libraries Chapter

DRAFT

SysML Partners (www.sysml.org)

American Systems Corporation
ARTISAN Software Tools*
BAE SYSTEMS
The Boeing Company
Ceira Technologies
Deere & Company
EADS Astrium GmbH
EmbeddedPlus Engineering
Eurostep Group AB
Georgia Institute of Technology
Gentleware AG
I-Logix*
International Business Machines*
International Council on Systems Engineering
Israel Aircraft Industries
Lockheed Martin Corporation
Mentor Graphics
Motorola*
National Aeronautics and Space Administration
National Institute of Standards and Technology
Northrop Grumman
oose.de Dienstleistungen für innovative Informatik GmbH
PivotPoint Technology Corporation
Raytheon Company
Structured Software Systems Limited
Telelogic AB*
THALES*
Vitech Corporation

* Submitter to OMG UML for Systems Engineering RFP

COPYRIGHT NOTICE

© 2003-2005 American Systems Corporation
© 2003-2005 ARTISAN Software Tools
© 2003-2005 BAE SYSTEMS
© 2003-2005 The Boeing Company
© 2003-2005 Ceira Technologies
© 2003-2005 Deere & Company
© 2003-2005 EADS Astrium GmbH
© 2003-2005 EmbeddedPlus Engineering
© 2003-2005 Eurostep Group AB
© 2003-2005 Gentleware AG
© 2003-2005 I-Logix, Inc.
© 2003-2005 International Business Machines
© 2003-2005 Israel Aircraft Industries
© 2003-2005 Lockheed Martin Corporation
© 2003-2005 Motorola, Inc.
© 2003-2005 Northrop Grumman
© 2003-2005 oose.de Dienstleistungen für innovative Informatik GmbH
© 2003-2005 PivotPoint Technology Corporation
© 2003-2005 Raytheon Company
© 2003-2005 Telelogic AB
© 2003-2005 THALES

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

This document describes a proposed language specification developed by an informal partnership of vendors and users, with input from additional reviewers and contributors. This document does not represent a commitment to implement any portion of this specification in any company's products. See the full text of this document for additional disclaimers and acknowledgments. The information contained in this document is subject to change without notice.

The specification proposes to customize the Unified Modeling Language (UML) specification of the Object Management Group (OMG) to address the requirements of Systems Engineering. These include many of the requirements requested by the UML for Systems Engineering RFP, OMG document number ad/03-03-41. This document includes references to and excerpts from the *UML 2.0 Superstructure Specification* (OMG document number ptc/2004-10-02) and *UML 2.0 Infrastructure Specification* (Final Adopted Specification; OMG document number ptc/2003-09-15) with copyright holders and conditions as noted in those documents.

LICENSES

Redistribution and use of this specification, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of this specification must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The Copyright Holders listed in the above copyright notice may not be used to endorse or promote products derived from this specification without specific prior written permission.
- All modified versions of this specification must include a prominent notice stating how and when the specification was modified.

THIS SPECIFICATION IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN

NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SPECIFICATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

TRADEMARKS

Systems Modeling Language and SysML, which are used to identify this specification, are not usable as trademarks since SysML Partners has established their usage to identify this specification without any trademark status or restriction. Organizations that wish to establish trademarks related to this specification should distinguish them somehow from SysML and Systems Modeling Language, for example by adding a unique prefix (e.g., OMG SysML).

Unified Modeling Language and UML are trademarks of the OMG. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

19 Profiles & Model Libraries

Editorial Comment: This SysML chapter draft is provided as an addendum to the SysML v. 0.9 draft (10 January 2005), submitted to the OMG as document# ad/05-01-03. It is intended to be a replacement for Chapter 19: *Profiles*, which was only a placeholder in v. 0.9. As with the SysML v. 0.9 draft, any review feedback regarding this chapter draft should be sent to the following public mailing list: SysMLforum@googlegroups.com.

19.1 Overview

The Profiles package specifies mechanisms that allow metaclasses from existing metamodels to be extended so that they can be adapted for different purposes, such as customizing SysML for different platforms or domains. The Profiles mechanism is intended to be architecturally compatible with the OMG UML 2.0 and Meta Object Facility (MOF) specifications.

Profiles cannot only be used to extend SysML, they can also be used to restrict the language by selecting the subset of the base metamodel that is required for the specific domain. For example, SysML does not require all of the UML metamodel. The Language Architecture chapter describes the subset of UML that is included in SysML.

The Usage Examples section provides guidance both on how to use existing profiles and how to create new profiles. In addition, the examples provide guidance on the use of model libraries.

19.2 Diagram elements

Table 1. Graphical nodes used in profile definition

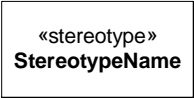


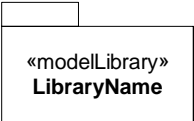
<i>NODE NAME</i>	<i>CONCRETE SYNTAX</i>	<i>ABSTRACT SYNTAX REFERENCE</i>	<i>COMPLIANCE</i>
Stereotype		UML::Profiles::Stereotype	Basic
Metaclass		UML::Profiles::Class	Basic
Profile		UML::Profiles::Profile	Basic
Model Library		UML::StandardProfileL1	Basic

Table 2. Graphical paths used in profile definition

<i>PATH NAME</i>	<i>CONCRETE SYNTAX</i>	<i>ABSTRACT SYNTAX REFERENCE</i>	<i>COMPLIANCE</i>

Table 2. Graphical paths used in profile definition

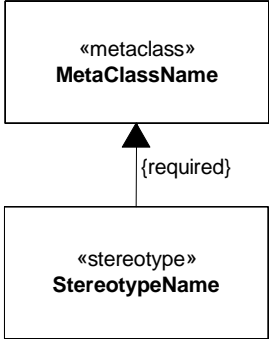
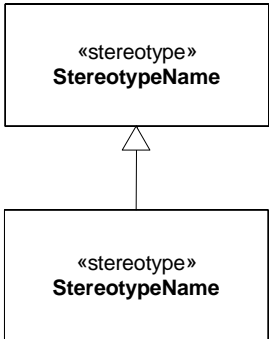
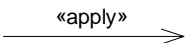
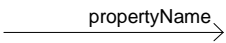
<p>Extension</p>	 <pre> classDiagram class MetaClassName["«metaclass» MetaClassName"] class StereotypeName["«stereotype» StereotypeName"] StereotypeName -- > MetaClassName : {required} </pre>	<p>UML::Profiles::Extension</p>	<p>Basic</p>
<p>Generalization</p>	 <pre> classDiagram class StereotypeName1["«stereotype» StereotypeName"] class StereotypeName2["«stereotype» StereotypeName"] StereotypeName2 -- > StereotypeName1 </pre>	<p>Infrastructure::Constructs::Generalization</p>	<p>Basic</p>
<p>ProfileApplication</p>	 <pre> classDiagram class ProfileApplication["«apply»"] ProfileApplication --> </pre>	<p>UML::Profiles::ProfileApplication</p>	<p>Basic</p>
<p>Unidirectional Association</p>	 <pre> classDiagram class UnidirectionalAssociation["propertyName"] UnidirectionalAssociation --> </pre>	<p>UML::Constructs::Association</p>	<p>Basic</p>

Table 3. Graphical conventions used in profile display

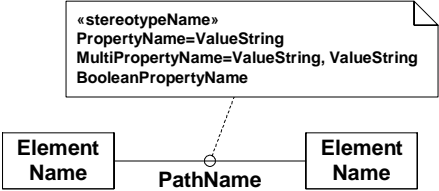
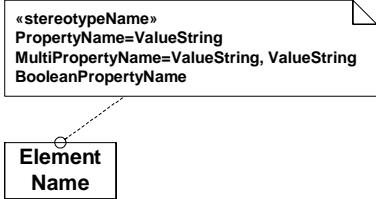
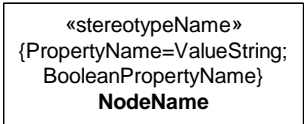
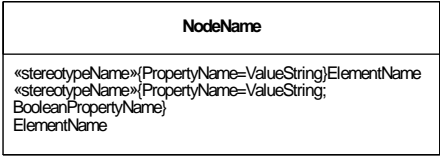
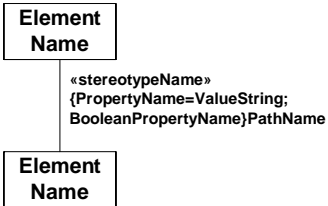
NODE NAME	CONCRETE SYNTAX	ABSTRACT SYNTAX REFERENCE	COMPLIANCE
Model Element		UML::Kernel::Element	Basic
Model Element		UML::Kernel::Element	Basic
Model Element		UML::Kernel::Element	Basic
Model Element		UML::Kernel::Element	Basic
Model Element		UML::Kernel::Element	Basic

Table 3. Graphical conventions used in profile display

NODE NAME	CONCRETE SYNTAX	ABSTRACT SYNTAX REFERENCE	COMPLIANCE
Model Element	<div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: fit-content;"> <p style="text-align: center;">«stereotypeName» NodeName</p> <hr style="border: 0; border-top: 1px solid black; margin: 2px 0;"/> <p>«stereotypeName» PropertyName=ValueString MultiPropertyName=ValueString, ValueString BooleanPropertyName</p> </div>	UML::Kernel::Element	Basic

Note – In the above table, boolean properties can alternatively be displayed as BooleanPropertyName=[True|False].

19.3 Package Structure

SysML does not add any new abstract syntax and so does not require an additional package.

19.4 UML extensions

19.4.1 Metaclass Extensions

None.

19.4.2 Diagram extensions

19.4.2.1 Stereotype

The values of a stereotype that has been applied to a model element can be shown in one of three ways:

- -As part of a comment symbol tied to the symbol representing the model element
- -In compartments of a graphic node representing the model element.
- -Above the name string within a graphic node or before the name string otherwise. Note that a restricted form of this notational option is simply just to show the stereotype name in guillemets.

In the case where a compartment or comment symbol is used, the user may elect to show the stereotype name in guillemets before the name string in addition to in the compartment or comment.

The values of the stereotype properties are displayed as name/value pairs, thus:

`<namestring>'='<valuestring>`

If a stereotype property is multi-valued then the valuestring is displayed as a comma-separated list:

`<valuestring>.:=<value>{'<value>'}`

Certain values have special display rules:

- As an alternative to a name/value pair, when displaying the values of boolean properties diagrams may use the convention that if the *namestring* is displayed then the value is True, otherwise the value is False;
- If the value is the name of a NamedElement then optionally its qualifiedName can be used.

If compartments are used to display stereotype values then an additional compartment is required for each applied stereotype whose values are to be displayed. Each such compartment is optionally headed by the name of the applied stereotype in guillemets. Any graphic node that is not displayed as an icon may have these compartments. Graphic nodes corresponding to the following SysML/UML concepts may have these compartments: any Classifier shown as a Class Node; Property (and subclasses); a CallBehaviorAction, CallOperation Action, CentralBufferNode (or subclasses) or ActivityParameterNode.

Editorial Comment: In the next SysML draft revision the list above should be replaced by a reference to specific SysML diagram elements.

Within a comment symbol, or if displayed before/above the symbol's namestring, the values from a specific stereotype are optionally preceded with the name of the applied stereotype within a pair of guillemets, which is useful if values of more than one applied stereotype should be shown.

When displayed in compartments or comment symbol at most one name/value pair can appear on a single line. When displayed above/before a namestring the name/value pairs are separated by semicolons and all pairs for a given stereotype are enclosed in braces.

In Figure 12, a simple stereotype *Clock* is defined to be applicable at will (dynamically) to instances of the metaclass *Class* and describes a clock software component for an embedded software system. It has description of the operating system version supported, an indication of whether it is compliant to the POSIX operating system standard and a reference to the operation that starts the clock.

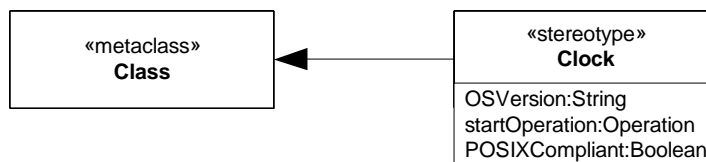


Figure 1 - Defining a stereotype

Figure 2 shows how the stereotype *Clock*, as defined in Figure 12, is applied to a class called *StopWatch*.



Figure 2 - Using a stereotype

When, two stereotypes, *Clock* and *Creator*, are applied to the same model element, as is shown in Figure 3, the attribute values of each of the applied stereotypes can be shown in a comment symbol attached to the model element.

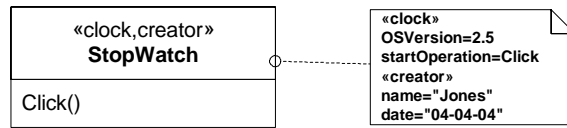


Figure 3 - Using stereotypes and showing values

Finally, the two alternate notational forms are shown..

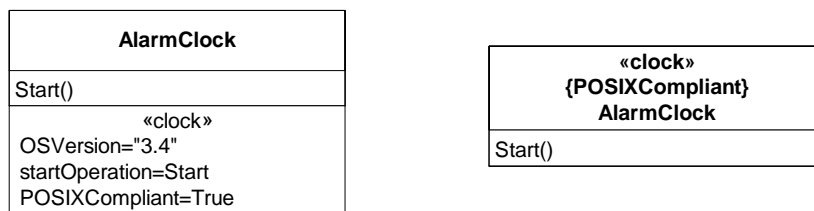


Figure 4 - Other notational forms for showing values

In this case, **AlarmClock** is valid for OS version 3.4, is POSIX-compliant and has a starting operation called `Start`. The compartment form of notation is shown on the left and the in-symbol form on the right (note that not all properties of **AlarmClock** are shown on the right. Note that multiple stereotypes can be shown using these alternates also, either as multiple compartments, or by grouping the values for a given stereotype after its stereotype indication.

19.5 Compliance levels

Elements have the compliance levels as indicated in the diagram elements table above.

19.6 Usage examples

19.6.1 Defining a Profile

Editorial Comment: Include an example of importing from a reference metamodel using the `<<metamodel>>` stereotype using a new "strict" attribute.

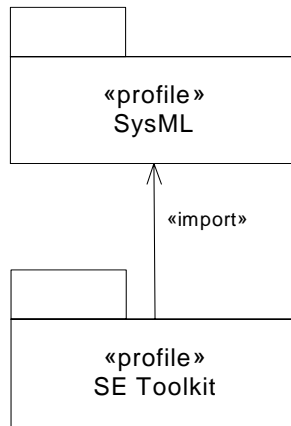


Figure 5 - Definition of a profile

In this example, the modeler has created a new profile called SE Toolkit, which imports the SysML profile, so that it can build upon the stereotypes it contains. The SE Toolkit can extend those metaclasses from UML that the SysML profile imports.

19.6.2 Adding Stereotypes to a Profile

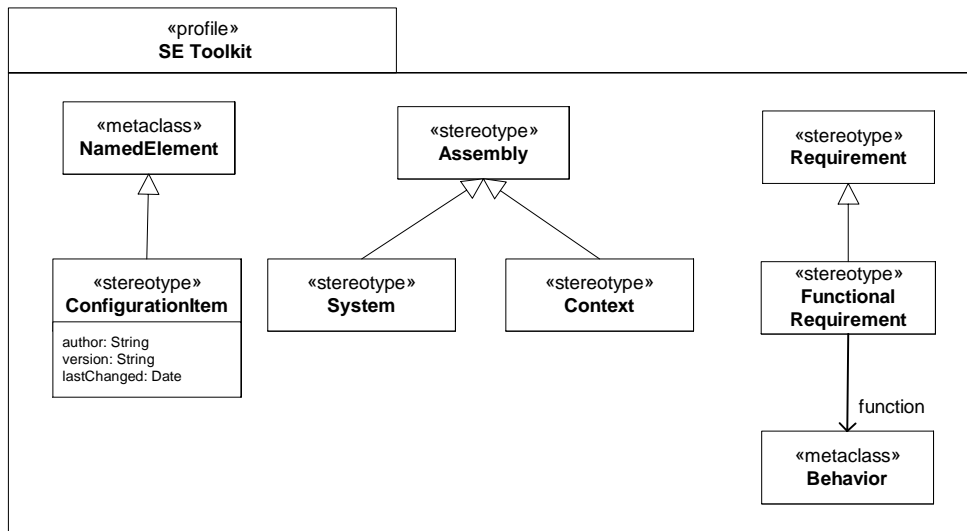


Figure 6 - Profile Contents

In SE Toolkit, both the mechanisms for adding new stereotypes are used. The first, exemplified by configurationItem, is called an extension, shown by a line with a filled triangle; this relates a stereotype to a reference (or base) class, in this case NamedElement from UML and adds new properties that every NamedElement stereotyped by configurationItem must have. NamedElement is an abstract class in UML so it is its subclasses that can have the stereotype applied. The second mechanism is demonstrated by the system and context stereotypes which are sub-stereotypes of an existing SysML stereotype, assembly; sub-stereotypes inherit any properties of their super-stereotype (in this case none) and also extend the same base class or

classes. Note that TypedElements whose type is extended by «system» do not display the «system» stereotype; this also applies to InstanceSpecifications. Any notational conventions of this have to be explicitly specified in a diagram extension.

T

There is also an example of how stereotypes (in this case FunctionalRequirement) can have unidirectional associations to metaclasses in the reference metamodel (in this case Behavior).

Editorial Comment: Include examples where: 1) two base classes are extended by the same stereotype; 2) a stereotype property is typed by a metaclass; and 3) subclassing stereotyped class and subclassing stereotype are constricted.

19.6.3 Defining a Model Library that uses a Profile

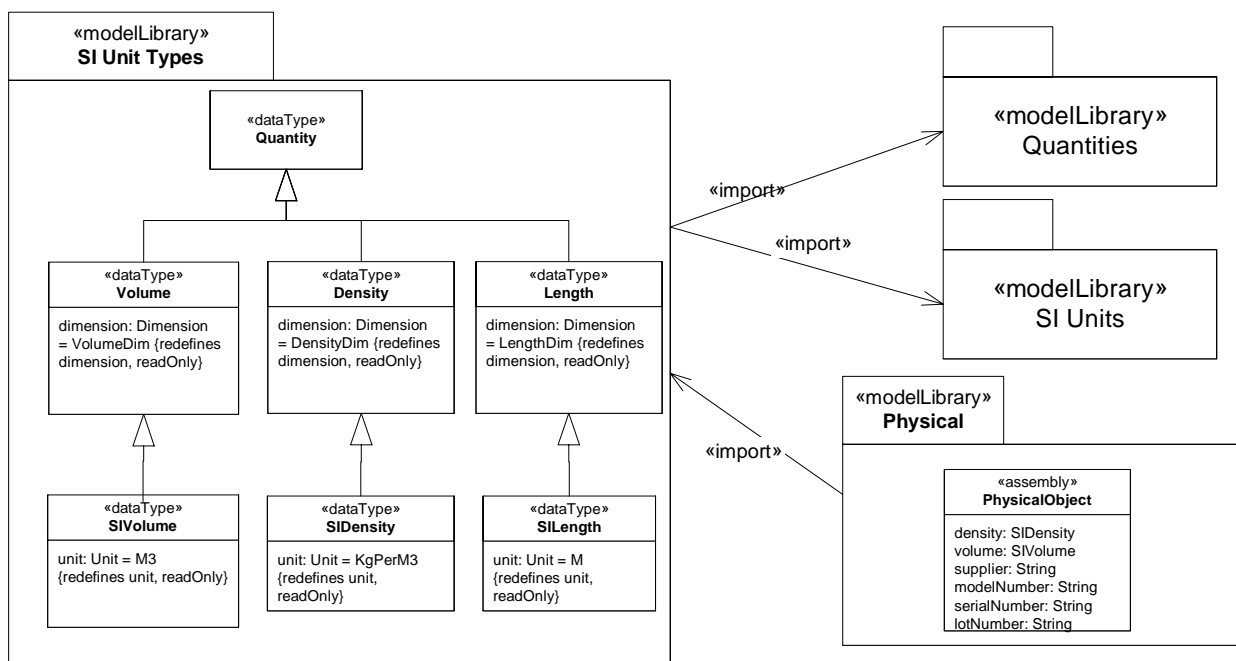


Figure 7 Two model libraries

The model library SI Unit Types imports the model libraries Quantities and SI Units from SysML, so that it can use model elements from them in its own definition. It defines subtypes of Quantity (as subset of which are shown here) that can be used when property values are measured in SI Units. A further model library, Physical, imports the SI Unit types so that it can define properties that have those types. One model element, PhysicalObject is shown, an assembly that can be used as a super-type for an physical object.

19.6.4 Guidance on whether to use a Stereotype or Class

This section provides guidance on when to use stereotypes. Stereotypes can be applied to any model element. Stereotyping a model element allows the model element to be identified with the «guillemet» notation. In addition, the stereotyped model element can have stereotype properties, and the stereotype can specify constraints on the model element.

The modeler must decide when to create a stereotype of a class versus when to specialize (subclass) the class. One reason is to be able to identify the class with the «guillemet» notation. In addition, the stereotype properties are different from properties of classes. Stereotype properties represent properties of the class that are not instantiated and therefore do not have a unique value for each instance of the class.

SE Toolkit::functionalRequirement, which extends Class through its superstereotype, Requirement, is an example where a stereotype is appropriate because every modeling element stereotyped by SE Toolkit::functionalRequirement has a reference to another modeling element. In another example, SE Toolkit::configurationItem defined above, which applies to classes amongst other concepts, is a stereotype because its properties characterise the author, version and last changed date of the modeling element themselves. One test of this is whether the new properties are inheritable; in this case author, version and last-changed date are not, because it is only those classes under configuration control that need the properties. To summarise, in the following circumstances a stereotype is appropriate:

- Where the model concept to be extended is not a class or class-based;
- Where the extensions include properties that reference other model elements;
- Where the extensions include properties that describe modeling data, not system data;

An example where a class is more appropriate is PhysicalObject from Figure 7 - in this case, the properties density and volume, and the component numbers, have distinct values for each system element described by the class, and are inherited by every subclass of PhysicalObject..

19.6.5 Using a Profile

Editorial Comment: [Show how to apply a profile to a model](#)

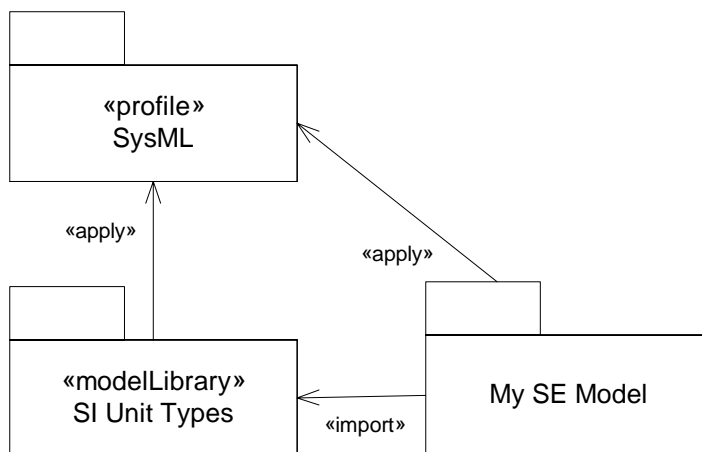


Figure 8 - A model with applied profile and imported model library

My SE Model is a system engineering model that needs to use stereotypes from SysML. It therefore needs to have the SysML profile applied to it. In order to use the predefined SI units, it also needs to import the SI Unit Types model library. Having done this, elements in My SE Model can be extended by SysML stereotypes and types like SIVolume can be used to type properties.

19.6.6 Using a Stereotype

Editorial Comment: Discuss how stereotypes are used, and distinguish between stereotype subclasses and user subclasses.

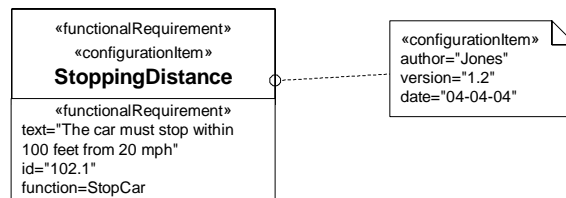


Figure 9 Using two stereotypes on a model element

StoppingDistance has two stereotypes applied, functionalRequirement, that identifies it as a requirement that is satisfied by a function, and configurationItem, which allows it to have configuration management properties. The modeler has provided values for all the newly available properties; those for criticalRequirement are shown in a compartment in the node symbol for StoppingDistance; those for configurationItem are shown in a separate note.

19.6.7 Using a Model Library Element

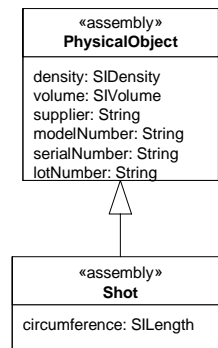


Figure 10 - Using model library elements

Editorial Comment: Show use and explain why this should be a model element, rather than a stereotype. Add Physical Object to a model library, and SLength to SI Unit Types.

Model library elements can be used just like any other model element of the same type. In this case, Shot is a specialisation of PhysicalObject from the Physical model library. It adds a new property, circumference, of type SLength to measure the circumference of the (spherical) shot.

