



BSI Standards Publication

Information technology — Object management group systems modeling language (OMG SysML)

National foreword

This British Standard is the UK implementation of ISO/IEC 19514:2017.

The UK participation in its preparation was entrusted to Technical Committee ICT/-/1, Information systems co-ordination.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2018
Published by BSI Standards Limited 2018

ISBN 978 0 580 92392 0

ICS 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 March 2018.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

Information technology — Object management group systems modeling language (OMG SysML)

*Technologies de l'information — Langage de modélisation de systèmes
OMG (OMG SysML)*



Reference number
ISO/IEC 19514:2017(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2017



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Table of Contents

FOREWORD	xix
INTRODUCTION	xx
1 Scope	1
1.1 General	1
2 Normative References	1
3 Additional Information	2
3.1 Relationships to Other Standards	2
3.2 How to Read this International Standard	2
3.2.1 Organization	3
3.3 Acknowledgments	4
4 Language Architecture	7
4.1 General	7
4.2 Design Principles	10
4.3 Architecture	10
4.4 Extension Mechanisms	13
4.5 SysML Diagrams	13
5 Conformance	15
5.1 Overview	15
5.2 Conformance Types	15
6 Language Formalism	17
6.1 Levels of Formalism	17
6.2 Clause Structure	17
6.2.1 Overview	17
6.2.2 Diagram Elements	17
6.2.3 UML Extensions	17
6.2.4 Usage Examples	18
6.3 Conventions and Typography	18
STRUCTURAL CONSTRUCTS	19

7 Model Elements	21
7.1 Overview	21
7.1.1 View and Viewpoint	21
7.2 Diagram Elements	22
7.3 UML Extensions	25
7.3.1 Diagram Extensions	25
7.3.1.1 UML Diagram Elements not Included in SysML	25
7.3.2 Stereotypes	26
7.3.2.1 Conform	26
7.3.2.2 ElementGroup	27
7.3.2.3 Expose	28
7.3.2.4 Problem	28
7.3.2.5 Rationale	29
7.3.2.6 Stakeholder	29
7.3.2.7 View	29
7.3.2.8 Viewpoint	30
7.4 Usage Examples	30
8 Blocks	33
8.1 Overview	33
8.2 Diagram Elements	34
8.2.1 Block Definition Diagram	34
8.2.2 Internal Block Diagram	40
8.3 UML Extensions	42
8.3.1 Diagram Extensions	42
8.3.1.1 Block Definition Diagram	42
8.3.1.2 Internal Block Diagram	44
8.3.1.3 UML Diagram Elements not Included in SysML Block Definition Diagrams	46
8.3.1.4 UML Diagram Elements not Included in SysML Internal Block Diagrams	46
8.3.2 Stereotypes	47
8.3.2.1 AdjunctProperty	49
8.3.2.2 Binding Connector	50
8.3.2.3 Block	51
8.3.2.4 Bound Reference	53
8.3.2.5 ClassifierBehaviorProperty	54
8.3.2.6 ConnectorProperty	54
8.3.2.7 DirectedRelationshipPropertyPath	55
8.3.2.8 DistributedProperty	56
8.3.2.9 ElementPropertyPath	56
8.3.2.10 EndPathMultiplicity	56
8.3.2.11 NestedConnectorEnd	57
8.3.2.12 ParticipantProperty	57
8.3.2.13 PropertySpecificType	58
8.3.2.14 ValueType	58
8.3.3 Model Libraries	59

8.3.3.1 Package PrimitiveValueTypes	59
8.3.3.2 Package UnitAndQuantityKind	60
8.4 Usage Examples	62
8.4.1 Wheel Hub Assembly	62
8.4.2 Example Value Type Definitions	64
8.4.3 Design Configuration for SUV EPA Fuel Economy Test	65
8.4.4 Water Delivery	65
8.4.5 Constraining Decomposition	65
8.4.6 Units and Quantity Kinds	67
9 Ports and Flows	71
9.1 Overview	71
9.1.1 Ports	71
9.1.2 Flow Properties, Provided and Required Features, and Nested Ports	71
9.1.3 Proxy Ports and Full Ports	71
9.1.4 Item Flows	72
9.1.5 Deprecation of Flow Ports and Flow Specifications	72
9.2 Diagram Elements	73
9.2.1 Block Definition Diagram	73
9.2.2 Internal Block Diagram	76
9.3 UML Extensions	78
9.3.1 Diagram Extensions	78
9.3.1.1 DirectedFeature	78
9.3.1.2 FlowProperty	78
9.3.1.3 FullPort	78
9.3.1.4 InvocationOnNestedPortAction	78
9.3.1.5 ItemFlow	78
9.3.1.6 Port	78
9.3.1.7 ProxyPort	79
9.3.1.8 TriggerOnNestedPort	79
9.3.2 Stereotypes	79
9.3.2.1 AcceptChangeStructuralFeatureEventAction	81
9.3.2.2 Block	82
9.3.2.3 ChangeStructuralFeatureEvent	82
9.3.2.4 DirectedFeature	82
9.3.2.5 FeatureDirection	83
9.3.2.6 FlowDirection	84
9.3.2.7 FlowProperty	84
9.3.2.8 FullPort	85
9.3.2.9 InterfaceBlock	86
9.3.2.10 InvocationOnNestedPortAction	86
9.3.2.11 ItemFlow	86
9.3.2.12 ProxyPort	87
9.3.2.13 TriggerOnNestedPort	88

9.4 Usage Examples	89
9.4.1 Ports with Required and Provided Features	89
9.4.2 Flow Ports and Item Flows	89
9.4.3 Ports with Flow Properties	90
9.4.4 Proxy and Full Ports	90
9.4.5 Association and Port Decomposition	91
9.4.6 Item Flow Decomposition	95
10 Constraint Blocks	97
10.1 Overview	97
10.2 Diagram Elements	98
10.2.1 Block Definition Diagram	98
10.2.2 Parametric Diagram	98
10.3 UML Extensions	99
10.3.1 Diagram Extensions	99
10.3.1.1 Block Definition Diagram	99
10.3.1.2 Parametric Diagram	101
10.3.2 Stereotypes	100
10.3.2.1 ConstraintBlock	101
10.4 Usage Examples	101
10.4.1 Definition of Constraint Blocks on a Block Definition Diagram	101
10.4.2 Usage of Constraint Blocks on a Parametric Diagram	101
BEHAVIORAL CONSTRUCTS	103
11 Activities	105
11.1 Overview	105
11.1.1 Control as Data	105
11.1.2 Continuous Systems	105
11.1.3 Probability	105
11.1.4 Activities as Blocks	106
11.1.5 Timelines	106
11.2 Diagram Elements	107
11.2.1 Activity Diagram	105
11.3 UML Extensions	114
11.3.1 Diagram Extensions	114
11.3.1.1 Activity	114
11.3.1.2 CallBehaviorAction	115
11.3.1.3 ControlFlow	116
11.3.1.4 ObjectNode, Variables, and Parameters	116
11.3.2 Stereotypes	117
11.3.2.1 Continuous	118

11.3.2.2 ControlOperator	119
11.3.2.3 Discrete	119
11.3.2.4 NoBuffer	119
11.3.2.5 Overwrite	120
11.3.2.6 Optional	120
11.3.2.7 Probability	120
11.3.2.8 Rate	121
11.3.3 Model Libraries	121
11.3.3.1 Package ControlValues	121
11.4 Usage Examples	122
12 Interactions	127
12.1 Overview	127
12.2 Diagram Elements	128
12.2.1 Sequence Diagram	128
12.3 UML Extensions	133
12.3.1 Diagram Extensions	133
12.3.1.1 Exclusion of Communication Diagram, Interaction Overview Diagram, and Timing Diagram	133
12.3.1.2 Interactions and Parameters	133
12.4 Usage Examples	134
12.4.1 Sequence Diagrams	134
13 State Machines	135
13.1 Overview	135
13.2 Diagram Elements	135
13.2.1 State Machine Diagram	135
13.3 UML Extensions	140
13.3.1 Diagram Extensions	140
13.3.1.1 State Machines and Parameters	140
13.4 Usage Examples	140
13.4.1 State Machine Diagram	140
14 Use Cases	141
14.1 Overview	141
14.2 Diagram Elements	142
14.2.1 Use Case Diagram	142
14.3 UML Extensions	143
14.4 Usage Examples	143
CROSSCUTTING CONSTRUCTS	145

15	Allocations	147
15.1	Overview.....	147
15.2	Diagram Elements	147
15.2.1	Representing Allocation on Diagrams.....	148
15.3	UML Extensions	149
15.3.1	Diagram Extensions	149
15.3.1.1	Tables	149
15.3.1.2	Allocate Relationship Rendering	149
15.3.1.3	Allocation Compartment Format	149
15.3.1.4	Allocation Callout Format	149
15.3.1.5	AllocatedActivityPartition Label	149
15.3.2	Stereotypes	150
15.3.2.1	Allocate(from Allocations)	150
15.3.2.2	AllocateActivityPartition(from Allocations)	151
15.4	Usage Examples	152
15.4.1	Behavior Allocation of Actions to Parts and Activities to Blocks	152
15.4.2	Allocate Flow	153
15.4.2.1	Allocating Structure	154
15.4.2.2	Automotive Example	154
15.4.3	Tabular Representation.....	155
16	Requirements	157
16.1	Overview.....	157
16.2	Diagram Elements	159
16.2.1	Requirement Diagram	159
16.3	UML Extensions	162
16.3.1	Diagram Extensions	162
16.3.1.1	Requirement Diagram	162
16.3.1.2	Requirement Notation	162
16.3.1.3	Requirement Property Callout Format	162
16.3.1.4	Requirements on Other Diagrams	162
16.3.1.5	Requirements Table	163
16.3.2	Stereotypes	164
16.3.2.1	Copy	164
16.3.2.2	DeriveReq	165
16.3.2.3	Refine	165
16.3.2.4	Requirement	165
16.3.2.5	TestCase	167
16.3.2.6	Satisfy	167
16.3.2.7	Trace	167
16.3.2.8	Verify	168
16.4	Usage Examples	168
16.4.1	Requirement Decomposition and Traceability	168
16.4.2	Requirements and Design Elements.....	169

16.4.3 Requirements Reuse	171
16.4.4 Verification Procedure (Test Case)	172
17 Profiles & Model Libraries	175
17.1 Overview	175
17.2 Diagram Elements	176
17.2.1 Profile Definition in Package Diagram	176
17.2.1.1 Extension	178
17.2.2 Stereotypes Used On Diagrams	178
17.2.2.1 StereotypeInNode	179
17.2.2.2 StereotypeInComment	180
17.2.2.3 StereotypeInCompartment	180
17.3 UML Extensions	180
17.4 Usage Examples	180
17.4.1 Defining a Profile	180
17.4.2 Adding Stereotypes to a Profile	181
17.4.3 Defining a Model Library that Uses a Profile	182
17.4.4 Guidance on Whether to Use a Stereotype or Class	183
17.4.5 Using a Profile	183
17.4.6 Using a Stereotype	184
17.4.7 Using a Model Library Element	184
ANNEXES	187
Annex A: Diagrams	189
Annex B: SysML Diagram Interchange	195
Annex C: Deprecated Elements	205
Annex D: Sample Problem	213
Annex E: Non-normative Extensions	251
Annex F: Requirements Traceability	319
Annex G: Model Interchange	321
Annex H: Legal Information	325

List of Figures

Figure 4.1 - Overview of SysML/UML Interrelationship.....	7
Figure 4.2 - SysML Extension of UML.....	11
Figure 4.3 - SysML Package Structure.....	12
Figure 4.4 - Non-normative Package Structure.....	13
Figure 7.1 - Stereotypes defined in package ModelElements.....	26
Figure 7.2 - Rationale and Problem examples.....	31
Figure 8.1 - Nested property reference.....	45
Figure 8.2 - Abstract syntax extensions for SysML blocks.....	47
Figure 8.3 - Abstract syntax extensions for SysML properties.....	47
Figure 8.4 - Abstract syntax extensions for SysML value types.....	47
Figure 8.5 - Abstract syntax extensions for SysML property paths.....	48
Figure 8.6 - Abstract syntax extensions for SysML connector ends.....	48
Figure 8.7 - Abstract syntax extensions for SysML property-specific types.....	48
Figure 8.8 - Abstract syntax extensions for SysML bound references.....	49
Figure 8.9 - Abstract syntax extensions for SysML adjunct properties and classifier behavior properties.....	49
Figure 8.10 - Model library for primitive value types.....	59
Figure 8.11 - Model library for Unit and QuantityKind.....	60
Figure 8.12 - Block diagram for the Wheel Package.....	63
Figure 8.13 - Internal Block Diagram for WheelHubAssembly.....	64
Figure 8.14 - Defining Value Types with units of measure from the International System of Units (SI).....	64
Figure 8.15 - Vehicle decomposition.....	65
Figure 8.16 - Vehicle internal structure.....	66
Figure 8.17 - Vehicle specialization.....	66
Figure 8.18 - Example of Unit, QuantityKind and ValueType definitions.....	67
Figure 8.19 - Instance-level view of the Unit, QuantityKind and ValueType definitions.....	68
Figure 8.20 - Example of equivalent Unit representations.....	68
Figure 8.21 - Instance-level representation of equivalent Unit definitions.....	69
Figure 9.1 - Port Stereotypes.....	79
Figure 9.2 - Stereotypes for Actions on Nested Ports.....	80
Figure 9.3 - Stereotypes for Property Value Change Events.....	80
Figure 9.4 - Provided and Required Features.....	80
Figure 9.5 - ItemFlow Stereotype.....	81
Figure 9.6 - Usage example of ports with provided and required features.....	89
Figure 9.7 Usage example of proxy and full ports.....	91
Figure 9.8 - Water Delivery association block.....	92
Figure 9.9 - Internal structure of Water Delivery association block.....	92
Figure 9.10 - Two views of Water Delivery connector within House block.....	93
Figure 9.11 - Specializations of Water Client in house example.....	93
Figure 9.12 - Plumbing association block.....	94
Figure 9.13 - Internal structure of Plumbing association block.....	94
Figure 9.14 - Water Delivery association block with internal Plumbing connector.....	94

Figure 9.15 - Usage example of item flows in internal block diagrams	95
Figure 9.16 - Usage example of item flow decomposition	96
Figure 9.17 - Usage example of item flow decomposition	96
Figure 10.1 - Stereotypes defined in SysML ConstraintBlocks package.....	100
Figure 11.1 - Block definition diagram with activities as blocks.....	115
Figure 11.2 - CallBehaviorAction notation.with behavior stereotype	115
Figure 11.3 - CallBehaviorAction notation.with action name	115
Figure 11.4 - Control flow notation.....	116
Figure 11.5 - Block definition diagram with activities as blocks associated with types of object nodes, variables, and parameters.....	116
Figure 11.6 - ObjectNode notation in activity diagrams	117
Figure 11.7 - ObjectNode notation in activity diagrams	117
Figure 11.8 - Abstract Syntax for SysML Activity Extensions	118
Figure 11.9 - Control values.....	121
Figure 11.10 - Continuous system example 1	123
Figure 11.11 - Continuous system example 2	124
Figure 11.12 - Continuous system example 3	124
Figure 11.13 - Example block definition diagram for activity decomposition	125
Figure 11.14 - Example block definition diagram for object node types.....	125
Figure 12.1 - Block definition diagram with interactions as blocks associated with used interactions and types of parameters.....	133
Figure 13.1 - Block definition diagram with state machines as blocks associated with submachines and types of parameters.....	140
Figure 15.1 - Abstract syntax extensions for SysML Allocation.....	150
Figure 15.2 - Abstract syntax expression for AllocatedActivityPartition.....	150
Figure 15.3 - Generic Allocation, including /from and /to association ends	152
Figure 15.4 - Behavior allocation.....	152
Figure 15.5 - Example of flow allocation from ObjectFlow to Connector	153
Figure 15.6 - Example of flow allocation from ObjectFlow to ItemFlow	153
Figure 15.7 - Example of flow allocation from ObjectNode to FlowProperty	154
Figure 15.8 - Example of Structural Allocation	154
Figure 15.9 - Allocation Matrix showing Allocation for Hybrid SUV Accelerate Example.....	155
Figure 16.1 - Non-normative Examples of Tabular Representations of Requirements	163
Figure 16.2 - Abstract Syntax for Requirements Stereotypes	164
Figure 16.3 - Requirements Derivation.....	169
Figure 16.4 - Links between requirements and design.....	170
Figure 16.5 - Requirement satisfaction in an internal block diagram	171
Figure 16.6 - Use of the copy dependency to facilitate reuse	171
Figure 16.7 - Linkage of a Test Case to a requirement: This figure shows the Requirement Diagram	172
Figure 16.8 - Linkage of a Test Case to a requirement: This figure shows the Test Case as a State Diagram.....	173
Figure 17.1 - Defining a stereotype.....	178
Figure 17.2 - Using a stereotype	179

Figure 17.3 - Using stereotypes and showing values.....	180
Figure 17.4 - Other notational forms for showing values	180
Figure 17.5 - Definition of a profile.....	181
Figure 17.6 - Profile Contents.....	181
Figure 17.7 - Two model libraries.....	182
Figure 17.8 - A model with applied profile and imported model library.....	183
Figure 17.9 - Using two stereotypes on a model element	184
Figure 17.10 - Using model library elements.....	184
Figure A.1 - SysML Diagram Taxonomy	190
Figure A.2 - Diagram Frame.....	191
Figure A.3 - Diagram Usages.....	193
Figure A.4 - Optional Form of Line Crossing.....	194
Figure C.1 - Deprecated Stereotypes	208
Figure D.1 - Establishing the User Model by Importing and Applying SysML Profile & Model Library (Package Diagram).....	214
Figure D.2 - Defining valueTypes and units to be Used in the Sample Problem	215
Figure D.3 - Establishing Structure of the User Model using Packages and Views (Package Diagram)	216
Figure D.4 - Establishing the Context of the Hybrid SUV System using a User-Defined Context Diagram. (Internal Block Diagram) Completeness of Diagram Noted in Diagram Description.....	217
Figure D.5 - Establishing Top Level Use Cases for the Hybrid SUV (Use Case Diagram).....	218
Figure D.6 - Establishing Operational Use Cases for “Drive the Vehicle” (Use Case Diagram).....	219
Figure D.7 - Elaborating Black Box Behavior for the “Drive the Vehicle” Use Case (Sequence Diagram).....	220
Figure D.8 - Finite State Machine Associated with “Drive the Vehicle” (State Machine Diagram).....	221
Figure D.9 - Black Box Interaction for “StartVehicle,” referencing White Box Interaction (Sequence Diagram).....	221
Figure D.10 - White Box Interaction for “StartVehicle” (Sequence Diagram).....	222
Figure D.11 - Establishing HSUV Requirements Hierarchy (containment) - (Requirements Diagram).....	223
Figure D.12 - Establishing Derived Requirements and Rationale from Lowest Tier of Requirements Hierarchy (Requirements Diagram).....	224
Figure D.13 - Acceleration Requirement Relationships (Requirements Diagram)	225
Figure D.14 - Requirements Relationships Expressed in Tabular Format (Table)	226
Figure D.15 - Defining the Automotive Domain (compare with Figure D.4) - (Block Definition Diagram)	227
Figure D.16 - Defining Structure of the Hybrid SUV System (Block Definition Diagram)	227
Figure D.17 - Internal Structure of Hybrid SUV (Internal Block Diagram).....	228
Figure D.18 - Defining Structure of Power Subsystem (Block Definition Diagram).....	229
Figure D.19 - Internal Structure of the Power Subsystem (Internal Block Diagram).....	230
Figure D.20 - Blocks Typing Ports in the Power Subsystem (Block Definition Diagram)	230
Figure D.21 - Initially Defining Port Types with Flow Properties for the CAN Bus (Block Definition Diagram).....	231
Figure D.22 - Consolidating Connectors into the CAN Bus. (Internal Block Diagram).....	232
Figure D.23 - Elaborating Definition of Fuel Flow. (Block Definition Diagram).....	232
Figure D.24 - Defining Fuel Flow Constraints (Parametric Diagram)	233
Figure D.25 - Detailed Internal Structure of Fuel Delivery Subsystem (Internal Block Diagram)	234
Figure D.26 - Defining Analyses for Hybrid SUV Engineering Development (Block Definition Diagram)	235
Figure D.27 - Establishing a Performance View of the User Model (Package Diagram)	236
Figure D.28 - Defining Requirements and VnV viewpoints (Package Diagram).....	237

Figure D.29 - Requirements and VnV views exposing elements from the model (Package Diagram).....	238
Figure D.30 - The Requirements and VnV views with supporting views (Package Diagram)	239
Figure D.31 - Defining Measures of Effectiveness and Key Relationships (Parametric Diagram).....	240
Figure D.32 - Establishing Mathematical Relationships for Fuel Economy Calculations (Parametric Diagram).....	241
Figure D.33 - Straight Line Vehicle Dynamics Mathematical Model (Parametric Diagram).....	242
Figure D.34 - Defining Straight-Line Vehicle Dynamics Mathematical Constraints (Block Definition Diagram).....	243
Figure D.35 - Results of Maximum Acceleration Analysis (Timing Diagram).....	244
Figure D.36 - Behavior Model for “Accelerate” Function (Activity Diagram).....	245
Figure D.37 - Decomposition of “Accelerate” Function (Block Definition diagram).....	246
Figure D.38 - Detailed Behavior Model for “Provide Power” (Activity Diagram) Note hierarchical consistency with Figure D.36.....	247
Figure D.39 - Flow Allocation to Power Subsystem (Internal Block Diagram).....	248
Figure D.40 - Tabular Representation of Allocation from “Accelerate” Behavior Model to Power Subsystem (Table).....	248
Figure D.41 - Special Case of Internal Block Diagram Showing Reference to Specific Properties (serial numbers).....	249
Figure E.1 - Example activity with «effbd» stereotype applied.....	253
Figure E.2 - Example activity with «streaming» and «nonStreaming» stereotypes applied to subactivities	253
Figure E.3 - Example extensions to Requirement.....	256
Figure E.4 - Example Parametric Diagram using Stereotypes for Measures of Effectiveness	257
Figure E.5 - QUDV Concepts diagram	259
Figure E.6 - QUDV Units diagram	260
Figure E.7 - QUDV Quantity Kinds diagram	260
Figure E.8 - Base Unit and Quantity Kinds of the SI and ISQ respectively	278
Figure E.9 - Example of a derived unit and derived quantity kind	278
Figure E.10 - Spring Length Example	279
Figure E.11 - Model libraries of SysML Quantity Kinds and Units for the covered content of ISO 80000 parts 3,4,5,6,7,9,10 and 13	280
Figure E.12 - Organization of the definitions of units and quantities from the normative parts of ISO 80000 covered in SysML 1.4, which includes all the normative content of parts 3,4,5,6; the subset of parts 7,9,10 corresponding to the content from SysML 1.3 and the subset of part 13 pertaining to commonly used units of information. Parts 8,11 and 12 are not covered because none of their units and quantities were referenced in previous versions of SysML nor in the summary tables in ISO 80000-1	281
Figure E.13 - Content relationships for the systems of units and quantities in from the different parts of ISO 80000 in relation to ISO 80000 as a whole and to the International System of Units (SI) and quantities (ISQ)	282
Figure E.14 - Table 1 (from ISO 80000-1) SI base units for the ISQ base quantities	283
Figure E.15 - Table 2 (from ISO 80000-1) ISQ derived quantities and SI derived units with special names (1)	284
Figure E.16 - Table 2 (from ISO 80000-1) ISQ derived quantities and SI derived units with special names (2)	285
Figure E.17 - Table 2 (from ISO 80000-1) ISQ derived quantities and SI derived units with special names (3)	286
Figure E.18 - Table 3 (from the SI brochure) SI derived units with special names and symbols.....	287
Figure E.19 - Constant numbers used throughout the SysML ISO 80000 library.	289

Figure E.20 - Example of value type definitions for a quantity and applicable units and prefixed units.....290

Figure E.21 - Basic distribution stereotypes 316

Figure E.22 - Distribution Example.....317

Figure G.1 - SysML/AP233 Data Overlaps.....322

List of Tables

Table 4.1 - UML 2 metaclasses excluded from the UML4SysML subset	8
Table 4.2 - UML 2 metaclasses and datatypes included in the UML4SysML subset	9
Table 4.3 - SysML stereotypes, blocks, valuetypes, and datatypes	10
Table 7.1 - Graphical nodes defined by ModelElements package	22
Table 7.2 - Graphical paths defined by ModelElements package	24
Table 8.1 - Graphical nodes defined in Block Definition diagrams	34
Table 8.2 - Graphical paths defined by in Block Definition diagrams	37
Table 8.3 - Graphical nodes defined in Internal Block diagrams	40
Table 8.4 - Graphical paths defined in Internal Block diagrams	41
Table 9.1 - Graphical nodes defined in Block Definition diagrams	73
Table 9.2 - Graphical nodes defined in Internal Block diagrams	76
Table 10.1 - Graphical nodes defined in Block Definition diagrams	98
Table 10.2 - Graphical nodes defined in Parametric diagrams	99
Table 11.1 - Graphical nodes included in activity diagrams	107
Table 11.2 - Graphical paths included in activity diagrams	112
Table 11.3 - Other graphical elements included in activity diagrams	113
Table 12.1 - Graphical nodes included in sequence diagrams	128
Table 12.2 - Graphical paths included in sequence diagram	132
Table 12.3 - Other graphical elements included in sequence diagram	132
Table 13.1 - Graphical nodes included in state machine diagrams	135
Table 13.2 - Graphical paths included in state machine diagrams	139
Table 13.3 - Other graphical elements included in state machine diagram	139
Table 14.1 - Graphical nodes included in Use Case diagrams	142
Table 14.2 - Graphical paths included in Use Case diagrams	143
Table 15.1 - Extension to graphical nodes included in diagrams	148
Table 16.1 - Graphical nodes included in Requirement diagrams	159
Table 16.2 - Graphical paths included in Requirement diagrams	160
Table 17.1 - Graphical nodes used in profile definition	176
Table 17.2 - Graphical paths used in profile definition	177
Table 17.3 - Notations for Stereotype Use	178
Table 17.4 - Notations for Stereotype Use (continued)	179
Table B.1 - SysML Diagram Elements	201
Table C.1 - Graphical nodes defined in block definition diagrams	206
Table C.2 - Graphical nodes defined in internal block diagrams	207
Table E.1 - Addition stereotypes for EFFBDs	251
Table E.2 - Streaming options for activities	252
Table E.3 - Additional Requirement Stereotypes	254
Table E.4 - Requirement property enumeration types	255
Table E.5 - Stereotypes for Measures of Effectiveness	257

Table E.6 - The decimal and binary prefixes in scope of the International System of Units (SI) which uses the ISO 80000 system of units and its included systems of units such as ISO 80000-13	287
Table E.7 - Normative units in ISO 80000-3 (1 of 2)	291
Table E.8 - Normative units in ISO 80000-3 (2 of 2)	292
Table E.9 - Normative quantity kinds in ISO 80000-3 (1 of 2)	292
Table E.10 - Normative quantity kinds in ISO 80000-3 (2 of 2)	293
Table E.11 - Normative units in ISO 80000-4 (1 of 2).....	294
Table E.12 - Normative units in ISO 80000-4 (2 of 2)	295
Table E.13 - Normative quantity kinds in ISO 80000-4 (1 of 4)	296
Table E.14 - Normative quantity kinds in ISO 80000-4 (2 of 4)	297
Table E.15 - Normative quantity kinds in ISO 80000-4 (3 of 4)	298
Table E.16 - Normative quantity kinds in ISO 80000-4 (4 of 4)	299
Table E.17 - Normative units in ISO 80000-5 (1 of 2)	300
Table E.18 - Normative units in ISO 80000-5 (2 of 2)	301
Table E.19 - Normative quantity kinds in ISO 80000-5 (1 of 5)	302
Table E.20 - Normative quantity kinds in ISO 80000-5 (2 of 5)	303
Table E.21 - Normative quantity kinds in ISO 80000-5 (3 of 5)	303
Table E.22 - Normative quantity kinds in ISO 80000-5 (4 of 5)	304
Table E.23 - Normative quantity kinds in ISO 80000-5 (5 of 5)	305
Table E.24 - Normative units in ISO 80000-6 (1 of 5)	306
Table E.25 - Normative units in ISO 80000-6 (2 of 5)	307
Table E.26 - Normative units in ISO 80000-6 (3 of 5)	307
Table E.27 - Normative units in ISO 80000-6 (4 of 5)	308
Table E.28 - Normative units in ISO 80000-6 (5 of 5)	309
Table E.29 - Normative quantity kinds in ISO 80000-6 (1 of 4)	310
Table E.30 - Normative quantity kinds in ISO 80000-6 (2 of 4)	311
Table E.31 - Normative quantity kinds in ISO 80000-6 (3 of 4)	312
Table E.32 - Normative quantity kinds in ISO 80000-6 (4 of 4)	313
Table E.33 - Units in ISO 80000-7.....	314
Table E.34 - Quantity Kinds in ISO 80000-7.....	314
Table E.35 - Units in ISO 80000-9.....	314
Table E.36 - Quantity Kinds in ISO 80000-9.....	315
Table E.37 - Units in ISO 80000-10.....	315
Table E.38 - Quantity Kinds in ISO 80000-10.....	315
Table E.39 - Units in ISO 80000-13.....	315
Table E.40 - Quantity Kinds in ISO 80000-13.....	316
Table E.41 - Distribution Stereotypes	317

FOREWORD

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by the Object Management Group (OMG) and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

This document is related to:

- ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1995, *Information Technology - Open Distributed Processing - Reference Model: Foundations*
- ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1995, *Information Technology - Open Distributed Processing - Reference Model: Architecture*
- ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1997, *Information Technology - Open Distributed Processing - Interface Definition Language*

Apart from this Foreword, the text of this document is identical with that for the OMG specification for Systems Modeling Language, v1.4.1.

INTRODUCTION

The rapid growth of distributed processing has led to a need for a coordinating framework for this standardization and ITU-T Recommendations X.901-904 | ISO/IEC 10746, the Reference Model of Open Distributed Processing (RM-ODP) provides such a framework. It defines an architecture within which support of distribution, interoperability and portability can be integrated.

RM-ODP Part 2 (ISO/IEC 10746-2) defines the foundational concepts and modeling framework for describing distributed systems. The scopes and objectives of the RM-ODP Part 2 and the UML, while related, are not the same and, in a number of cases, the RM-ODP Part 2 and the UML specification use the same term for concepts which are related but not identical (e.g., interface). Nevertheless, a specification using the Part 2 modeling concepts can be expressed using UML with appropriate extensions (using stereotypes, tags, and constraints).

RM-ODP Part 3 (ISO/IEC 10746-3) specifies a generic architecture of open distributed systems, expressed using the foundational concepts and framework defined in Part 2. Given the relation between UML as a modeling language and Part 3 of the RM-ODP standard, it is easy to show that UML is suitable as a notation for the individual viewpoint specifications defined by the RM-ODP.

This International Standard for OMG Systems Modeling Language is a standard for the technology specification of an ODP system. It defines a technology to provide the infrastructure required to support functional distribution of an ODP system, specifying functions required to manage physical distribution, communications, processing and storage, and the roles of different technology objects in supporting those functions.

This International Standard defines a general-purpose language for systems engineering applications, called the OMG Systems Modeling Language (OMG SysML™). Throughout the rest of this International Standard the language will be referred to as SysML.

SysML supports the specification, analysis, design, verification, and validation of a broad range of complex systems. These systems may include hardware, software, information, processes, personnel, and facilities.

It is common practice for engineers to use a wide range of modeling languages, tools, and techniques on large systems projects. SysML is intended to unify diverse modeling languages used by systems engineers and can be used with a wide variety of discipline- and domain-specific modeling languages.

SysML reuses a subset of UML 2.5 and provides additional extensions needed to address the requirements in UML for SE. SysML uses the UML 2.5 extension mechanisms as further elaborated in Clause 17 as the primary mechanism to specify the extensions to UML 2.5. This revision of SysML relies on several new features incorporated into UML 2.5. Any use of the term “UML 2” or “UML” in this International Standard, unless otherwise noted, will refer to UML 2.5 in general and the UML 2.5 specification in particular.

Since SysML uses UML 2.5 as its foundation, systems engineers modeling with SysML and software engineers modeling with UML 2.5 will be able to collaborate on models of software-intensive systems. This will improve communication among the various stakeholders who participate in the systems development process and promote interoperability among modeling tools. It is anticipated that SysML will be customized to model domain-specific applications, such as automotive, aerospace, communication, and information systems.

Information technology - Object Management Group Systems Modeling Language (OMG SysML 1.4.1)

1 Scope

1.1 General

The purpose of this International Standard is to specify the Systems Modeling Language (SysML), a general-purpose modeling language for systems engineering. Its intent is to specify the language so that systems engineering modelers may learn to apply and use SysML; modeling tool vendors may implement and support SysML; and both can provide feedback to improve future versions. Note that a definition of “system” and “systems engineering” can be found in ISO/IEC/IEEE 15288.

SysML reuses a subset of UML 2 and provides additional extensions to satisfy the requirements of the language. This International Standard documents the language architecture in terms of the parts of UML 2 that are reused and the extensions to UML 2. The International Standard includes the concrete syntax (notation) for the complete language and specifies the extensions to UML 2. The reusable portion of the UML 2 standard is not included directly in the International Standard but is included by reference. The International Standard also provides examples of how the language can be used to solve common systems engineering problems.

SysML is designed to provide simple but powerful constructs for modeling a wide range of systems engineering problems. It is particularly effective in specifying requirements, structure, behavior, allocations, and constraints on system properties to support engineering analysis. The language is intended to support multiple processes and methods such as structured, object-oriented, and others, but each methodology may impose additional constraints on how a construct or diagram kind may be used. This version of the language supports most, but not all, of the requirements of the UML for Systems Engineering RFP, as shown in the Requirements Traceability referenced by Annex F. These gaps are intended to be addressed in future versions of SysML as indicated in the matrix.

The following sub clauses provide background information about this International Standard. Instructions for both systems engineers and tool vendors who read this International Standard are provided in “How to Read this International Standard.” The main body of this International Standard describes the normative technical content. The annexes include additional information to aid in understanding and implementation of this International Standard.

2 Normative References

The following normative documents contain provisions, which through reference in this text, constitute provisions of this International Standard. Subsequent amendments to, or revisions of, any of these publications do not apply.

- ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards, 7th Edition 2016
- ISO/IEC 10303-233:2012, STEP AP233, Product data representation and exchange: application protocol: Systems engineering
- ISO/IEC IEEE 15288:2015, Systems and software engineering - System life cycle process

- OMG Specification formal/2015-03-01, Unified Modeling Language, (UML) V2.5 (<http://www.omg.org/spec/UML/2.5/>)
- OMG Specification formal/2012-01-01, Object Constraint Language (OCL), V2.3.1 (<http://www.omg.org/spec/OCL/2.3.1/>)
- OMG Specification formal/2015-06-05, Meta Object Facility (MOF), V2.5 (<http://www.omg.org/spec/MOF/2.5/>)
- OMG Specification formal/2015-06-01, Diagram Definition, V1.1 (<http://www.omg.org/spec/DD/1.1/>)
- OMG Document ad/03-03-41, UML for Systems Engineering RFP (<http://www.omg.org/cgi-bin/doc?ad/2003-03-41>)
- OMG Document ormsc/2014-06-01, Model Driven Architecture (MDA) Guide rev. 2.0 (<http://www.omg.org/cgi-bin/doc?ormsc/2014-06-01>)
- VIM Edition 3 (VIM3), “International vocabulary of metrology - Basic and general concepts and associated terms (VIM)”, JCGM 200:2012 (JCGM 200:2008 with minor corrections)
- [Dybkaer-2010] Rene Dybkaer, “ISO terminological analysis of the VIM3 concepts of ‘quantity’ and ‘kind-of-quantity’”, Metrologia 47, (2010) 127-143

3 Additional Information

3.1 Relationships to Other Standards

SysML is defined as an extension of the OMG UML 2 standard. See Clause 2 for the current version of the UML 2 standard.

SysML is intended to be supported by two evolving interoperability standards including the OMG XMI 2 model interchange standard for UML 2 modeling tools and the ISO 10303 STEP AP233 data interchange standard for systems engineering tools. Overviews of the approach to model interchange and relevant references are included in Annex G.

SysML supports the OMG’s Model Driven Architecture (MDA) initiative by its reuse of UML and related standards. See OMG MDA Guide rev 2.0.

3.2 How to Read this International Standard

This International Standard is intended to be read by systems engineers so they may learn and apply SysML, and by modeling tool vendors so they may implement and support SysML.

Systems engineers should read the Overview, Diagram Elements, and Usage Examples sub clauses in each clause, and explore the UML Extensions as they see fit. Modeling tool vendors should read all clauses. In addition, systems engineers and vendors should read Annex D, “Sample Problem,” to understand how the language is applied to an example, and the document referenced by Annex F, “Requirements Traceability,” to understand how the requirements in the UML for SE RFP are satisfied by this International Standard.

Although the clauses are organized into logical groupings that can be read sequentially, this International Standard can be used for reference and may be read in a non-sequential manner.

3.2.1 Organization

This International Standard is organized as follows:

FOREWORD

INTRODUCTION

1 Scope

2 Normative References

3 Additional Information - includes Relationships to Other Standards, How to Read this International Standard, and Acknowledgments

4 Language Architecture - General Information, Design Principles, Architecture, and SsyML Diagrams

5 Conformance - General Information and Conformance Types

6 Language Formalism -

- Levels of Formalism
- Clause Structure
- Conventions and Typography

STRUCTURAL CONSTRUCTS

7 Model Elements - Refactors the kernel package from UML 2 and includes some extensions to provide some foundation capabilities for model management.

8 Blocks - Reuses and extends structured classes from UML 2 composite structures to provide the fundamental capability for describing system decomposition and interconnection, and to define different types of system properties including value properties with optional units of measure.

9 Ports and Flows - Provides the semantics for defining how blocks and parts interact through ports and how items flow across connectors.

10 Constraint Blocks - Defines how blocks are extended to be used on parametric diagrams. Parametric diagrams model a network of constraints on system properties to support engineering analysis, such as performance, reliability, and mass properties analysis.

BEHAVIORAL CONSTRUCTS

11 Activities - Defines the extensions to UML 2 activities, which represent the basic unit of behavior that is used in activity, sequence, and state machine diagrams. The activity diagram is used to describe the flow of control and flow of inputs and outputs among actions.

12 Interactions - Defines the constructs for describing message based behavior used in sequence diagrams.

13 State Machines - Describes the constructs used to specify state based behavior in terms of system states and their transitions.

14 Use Cases - Describes behavior in terms of the high level functionality and uses of a system, that are further specified in the other behavioral diagrams referred to above.

CROSSCUTTING CONSTRUCTS

15 Allocations

16 Requirements

17 Profiles & Model Libraries

ANNEXES

Annex A - Diagrams

Annex B - SysML Diagram Interchange

Annex C - Deprecated Elements

Annex D - Sample Problem

Annex E - Non-normative Extensions

Annex F - Requirements Traceability

Annex G - Model Interchange

Annex H - Legal Information

3.3 Acknowledgments

The following companies and organizations submitted or supported parts of the original version of this International Standard:

Industry

- American Systems Corporation
- BAE SYSTEMS
- Boeing
- Deere & Company
- EADS Astrium
- Eurostep
- Israel Aircraft Industries
- Lockheed Martin Corporation
- Motorola
- Northrop Grumman
- oose Innovative Informatik GmbH
- PivotPoint Technology
- Raytheon
- THALES

US Government

- NASA/Jet Propulsion Laboratory
- National Institute of Standards and Technology (NIST)
- DoD/Office of the Secretary of Defense (OSD)

Vendors

- ARTiSAN Software Tools
- Ceira Technologies
- EmbeddedPlus Engineering
- Gentleware
- IBM
- I-Logix
- Mentor Graphics
- Telelogic
- Structured Software Systems Limited
- Sparx Systems
- Vitech

Academia

- Georgia Institute of Technology

Liaisons

- Consultative Committee for Space Data Systems (CCSDS)
- Embedded Architecture and Software Technologies (EAST)
- International Council on Systems Engineering (INCOS)
- ISO STEP AP233
- Systems Level Design Language (SLDL) and Rosetta

The following persons were members of the team that designed and wrote this International Standard: Vincent Arnould, Laurent Balmelli, Ian Bailey, James Baker, Cory Bialowas, Conrad Bock, Carolyn Boettcher, Roger Burkhart, Murray Cantor, Bruce Douglass, Harald Eisenmann, Anders Ek, Brenda Ellis, Marilyn Escue, Sanford Friedenthal, Eran Gery, Hal Hamilton, Dwayne Hardy, James Hummel, Cris Kobryn, Michael Latta, John Low, Robert Long, Kumar Marimuthu, Alan Moore, Véronique Normand, Salah Obeid, Eldad Palachi, David Price, Bran Selic, Chris Sibbald, Joseph Skipper, Rick Steiner, Robert Thompson, Jim U'Ren, Tim Weikiens, Thomas Weigert, and Brian Willard.

In addition, the following persons contributed valuable ideas and feedback that significantly improved the content and the quality of this International Standard: Perry Alexander, Michael Chonoles, Mike Dickerson, Orazio Gurrieri, Julian Johnson, Jim Long, Henrik Lönn, Stephen Mellor, Dave Oliver, Jim Schier, Matthias Weber, Peter Shames, and the Georgia Institute of Technology research team including Manas Bajaj, Injoong Kim, Chris Paredis, Russell Peak, and Diego Tamburini. The SysML team also wants to acknowledge Pavel Hruby and his contribution by providing the Visio stencil for UML 2 that was adapted for most of the figures throughout this International Standard.

Additional organizations and individuals have contributed to further revisions of this International Standard, as completed by Finalization and Revision Task Forces listed under the OMG SysML Roadmap in the Preface above. Besides those already acknowledged above for their contributions to the original International Standard, the following additional persons have contributed to the Finalization or Revision Task Forces: Yves Bernard, Graham Bleakley, Fraser Chadburn, Chris Delp, Hans Peter de Koning, Sébastien Demathieu, Peter Denno, Huascar Espinoza, Allison Barnard Feeney, Sébastien Gérard, Matthew Hause, Kenn Hussey, Nerijus Jankevicius, Steve Jenkins, Robert Karban, Darren Kelly, Andreas Korff, Frédéric Mallet, Sam Mancarella, Julio Medina, Jishnu Mukerji, Chris Paredis, Axel Reichwein, Pete Rivett, Nicolas Rouquette, George Sawyer, Andrius Strazdauskas, Kritsana Uttamang, John Watson, Bernd Wenzel. Additional organizations who supported the work of contributors to the Finalization and Revision Task Forces, not already listed for the original submission above, include 88solutions, Adaptive, Atego, EADS, CEA LIST, European Southern Observatory, European Space Agency, Fachhochschule Vorarlberg, INRIA, Mathworks, TecNALIA Research and Innovation, No Magic, and Universidad de Cantabria.

4 Language Architecture

4.1 General

SysML reuses a subset of UML 2 and provides additional extensions needed to address requirements in the UML for Systems Engineering RFP. This International Standard documents the language architecture in terms of the parts of UML 2 that are reused and the extensions to UML 2. This clause explains design principles and how they are applied to define the SysML language architecture.

To visualize the relationship between the UML and SysML languages, consider the Venn diagram shown in Figure 4.1, where the sets of language constructs that comprise the UML and SysML languages are shown as the circles marked “UML” and “SysML,” respectively. The intersection of the two circles, shown by the region marked “UML reused by SysML,” indicates the UML modeling constructs that SysML reuses, called the UML4SysML subset. The region marked “SysML extensions to UML” in Figure 4.1 indicates the new modeling constructs defined for SysML that have no counterparts in UML, or which replace UML constructs. Note that there is also a part of UML 2 that is not required to implement SysML, which is shown by the region marked “UML not required by SysML.”

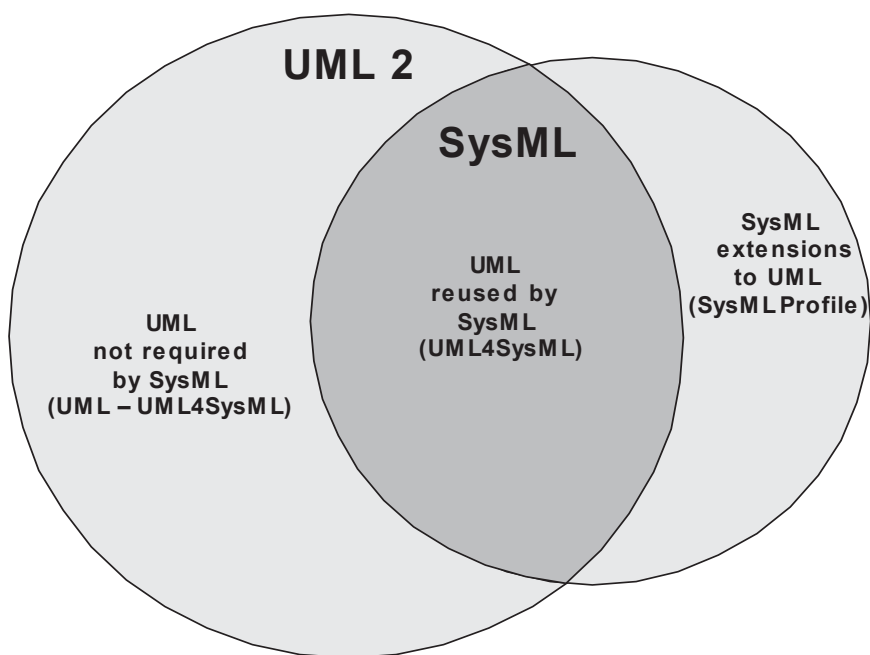


Figure 4.1- Overview of SysML/UML Interrelationship

Table 4.1 lists the metaclasses excluded from the UML4SysML subset. Table 4.2 lists the metaclasses and datatypes included in the UML4SysML subset. Table 4.3 lists the stereotypes, blocks, valuetypes, and datatypes included in SysML.

Table 4.1 - UML 2 metaclasses excluded from the UML4SysML subset

UML 2 metaclasses excluded from the UML4SysML subset
Artifact, ClassifierTemplateParameter, Collaboration, CollaborationUse, CommunicationPath, Component, ComponentRealization, ConnectableElementTemplateParameter, Deployment, DeploymentSpecification, Device, ExceptionHandler, ExecutionEnvironment, ExpansionNode, ExpansionRegion, Manifestation, Node, OperationTemplateParameter, ProtocolConformance, ProtocolStateMachine, ProtocolTransition, QualifierValue, ReadLinkObjectEndQualifierAction, RedefinableTemplateSignature, StringExpression, TemplateBinding, TemplateParameter, TemplateParameterSubstitution, TemplateSignature, UMLActivityDiagram, UMLAssociationEndLabel, UMLAssociationOrConnectorOrLinkShape, UMLAssociationOrConnectorOrLinkShapeKind, UMLBehaviorDiagram, UMLClassDiagram, UMLClassifierShape, UMLCompartment, UMLCompartmentableShape, UMLComponentDiagram, UMLCompositeStructureDiagram, UMLDeploymentDiagram, UMLDiagram, UMLDiagramElement, UMLDiagramWithAssociations, UMLEdge, UMLInteractionDiagram, UMLInteractionDiagramKind, UMLInteractionTableLabel, UMLKeywordLabel, UMLLabel, UMLMultiplicityLabel, UMLNameLabel, UMLNavigabilityNotationKind, UMLObjectDiagram, UMLPackageDiagram, UMLProfileDiagram, UMLRedefinesLabel, UMLShape, UMLStateMachineDiagram, UMLStateShape, UMLStereotypePropertyValueLabel, UMLStructureDiagram, UMLStyle, UMLTypedElementLabel, UMLUseCaseDiagram

Table 4.2 - UML 2 metaclasses and datatypes included in the UML4SysML subset

UML 2 metaclasses and datatypes included in the UML4SysML subset
Abstraction, AcceptCallAction, AcceptEventAction, Action, ActionExecutionSpecification, ActionInputPin, Activity, ActivityEdge, ActivityFinalNode, ActivityGroup, ActivityNode, ActivityParameterNode, ActivityPartition, Actor, AddStructuralFeatureValueAction, AddVariableValueAction, AggregationKind, AnyReceiveEvent, Association, AssociationClass, Behavior, BehaviorExecutionSpecification, BehavioralFeature, BehavioredClassifier, BroadcastSignalAction, CallAction, CallBehaviorAction, CallConcurrencyKind, CallEvent, CallOperationAction, CentralBufferNode, ChangeEvent, Class, Classifier, Clause, ClearAssociationAction, ClearStructuralFeatureAction, ClearVariableAction, CombinedFragment, Comment, ConditionalNode, ConnectableElement, ConnectionPointReference, Connector, ConnectorEnd, ConnectorKind, ConsiderIgnoreFragment, Constraint, Continuation, ControlFlow, ControlNode, CreateLinkAction, CreateLinkObjectAction, CreateObjectAction, DataStoreNode, DataType, DecisionNode, Dependency, DeployedArtifact, DeploymentTarget, DestroyLinkAction, DestroyObjectAction, DestructionOccurrenceSpecification, DirectedRelationship, Duration, DurationConstraint, DurationInterval, DurationObservation, Element, ElementImport, EncapsulatedClassifier, Enumeration, EnumerationLiteral, Event, ExecutableNode, ExecutionOccurrenceSpecification, ExecutionSpecification, Expression, Extend, Extension, ExtensionEnd, ExtensionPoint, Feature, FinalNode, FinalState, FlowFinalNode, ForkNode, FunctionBehavior, Gate, GeneralOrdering, Generalization, GeneralizationSet, Image, Include, InformationFlow, InformationItem, InitialNode, InputPin, InstanceSpecification, InstanceValue, Interaction, InteractionConstraint, InteractionFragment, InteractionOperand, InteractionOperatorKind, InteractionUse, Interface, InterfaceRealization, InterruptibleActivityRegion, Interval, IntervalConstraint, InvocationAction, JoinNode, Lifeline, LinkAction, LinkEndCreationData, LinkEndData, LinkEndDestructionData, LiteralBoolean, LiteralInteger, LiteralNull, LiteralReal, LiteralSpecification, LiteralString, LiteralUnlimitedNatural, LoopNode, MergeNode, Message, MessageEnd, MessageEvent, MessageKind, MessageOccurrenceSpecification, MessageSort, Model, MultiplicityElement, NamedElement, Namespace, ObjectFlow, ObjectNode, ObjectNodeOrderingKind, Observation, OccurrenceSpecification, OpaqueAction, OpaqueBehavior, OpaqueExpression, Operation, OutputPin, Package, PackageImport, PackageMerge, PackageableElement, Parameter, ParameterDirectionKind, ParameterEffectKind, ParameterSet, ParameterableElement, PartDecomposition, Pin, Port, PrimitiveType, PrimitiveTypes::Boolean, PrimitiveTypes::Integer, PrimitiveTypes::Real, PrimitiveTypes::String, PrimitiveTypes::UnlimitedNatural, PrimitiveValueTypes::Boolean, Profile, ProfileApplication, Property, Pseudostate, PseudostateKind, RaiseExceptionAction, ReadExtentAction, ReadIsClassifiedObjectAction, ReadLinkAction, ReadLinkObjectEndAction, ReadSelfAction, ReadStructuralFeatureAction, ReadVariableAction, Realization, Reception, ReclassifyObjectAction, RedefinableElement, ReduceAction, Region, Relationship, RemoveStructuralFeatureValueAction, RemoveVariableValueAction, ReplyAction, SendObjectAction, SendSignalAction, SequenceNode, Signal, SignalEvent, Slot, StartClassifierBehaviorAction, StartObjectBehaviorAction, State, StateInvariant, StateMachine, Stereotype, StructuralFeature, StructuralFeatureAction, StructuredActivityNode, StructuredClassifier, Substitution, TestIdentityAction, TimeConstraint, TimeEvent, TimeExpression, TimeInterval, TimeObservation, Transition, TransitionKind, Type, TypedElement, UnmarshallAction, Usage, UseCase, ValuePin, ValueSpecification, ValueSpecificationAction, Variable, VariableAction, Vertex, VisibilityKind, WriteLinkAction, WriteStructuralFeatureAction, WriteVariableAction

Table 4.3 - SysML stereotypes, blocks, valuetypes, and datatypes

SysML stereotypes, blocks, valuetypes, and datatypes
AcceptChangeStructuralFeatureEventAction, AdjunctProperty, Allocate, AllocateActivityPartition, BindingConnector, Block, BoundReference, ChangeStructuralFeatureEvent, ClassifierBehaviorProperty, Conform, ConnectorProperty, ConstraintBlock, Continuous, ControlOperator, ControlValue, Copy, DeriveReq, DirectedFeature, DirectedRelationshipPropertyPath, Discrete, DistributedProperty, ElementGroup, ElementPropertyPath, EndPathMultiplicity, Expose, FeatureDirection, FlowProperty, FullPort, InterfaceBlock, InvocationOnNestedPortAction, ItemFlow, NestedConnectorEnd, NoBuffer, Optional, Overwrite, ParticipantProperty, PrimitiveValueTypes::Boolean, PrimitiveValueTypes::Complex, PrimitiveValueTypes::Integer, PrimitiveValueTypes::Number, PrimitiveValueTypes::Real, PrimitiveValueTypes::String, Probability, Problem, PropertySpecificType, ProxyPort, Rate, Rationale, Refine, Requirement, Satisfy, Stakeholder, TestCase, Trace, TriggerOnNestedPort, ValueType, VerdictKind, Verify, View, Viewpoint

4.2 Design Principles

The fundamental design principles for SysML are:

- Requirements-driven - SysML is intended to satisfy the requirements of the UML for SE RFP.
- UML reuse - SysML reuses UML wherever practical to satisfy the requirements of the RFP, and when modifications are required, they are done in a manner that strives to minimize changes to the underlying language. Consequently, SysML is intended to be relatively easy to implement for vendors who support UML 2.
- UML extensions - SysML extends UML as needed to satisfy the requirements of the RFP. The primary extension mechanism is the UML 2 profile mechanism as further refined in Clause 17, “Profiles & Model Libraries.”
- Partitioning - The package is the basic unit of partitioning in this International Standard. The packages partition the model elements into logical groupings that minimize circular dependencies among them.
- Layering - SysML packages are specified as an extension layer to the UML metamodel.
- Interoperability - SysML inherits the XMI interchange capability from UML. SysML is also intended to be supported by the ISO 10303-233 data interchange standard to support interoperability among other engineering tools.

SysML provides three model libraries:

- PrimitiveValueTypes, see 8.3.3.1.
- UnitAndQuantityKind, see 8.3.3.2.
- ControlValues, see 11.3.3.

4.3 Architecture

The relationship between SysML and UML 2 is shown in Figure 4.2. SysML extends UML 2’s StandardProfile (see Clause 22 in the UML 2.5 specification) whose Trace and Refine stereotypes provide the basis for Requirement traceability in SysML (see Clause 16, “Requirements” in this International Standard).

Although SysML indirectly imports the UML 2 PrimitiveTypes library (see Clause 21 in the UML 2.5 specification) due to the transitivity of package import, SysML provides a PrimitiveValueTypes model library that systems engineers can extend via SysML's ValueType stereotype. In the remainder of this document, the unqualified references to Boolean, Integer, Real, and String should be interpreted as follows:

- In the context of the definition of a SysML Stereotype, the name refers to the definition of a UML::PrimitiveType in the UML 2 PrimitiveTypes library.
- Elsewhere, the name refers to the definition of a SysML::ValueType stereotype of UML::DataType in the SysML PrimitiveValueTypes library.

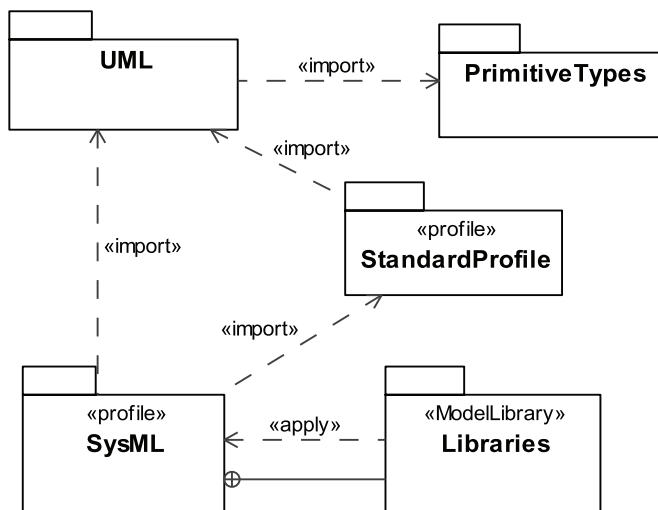


Figure 4.2 - SysML Extension of UML

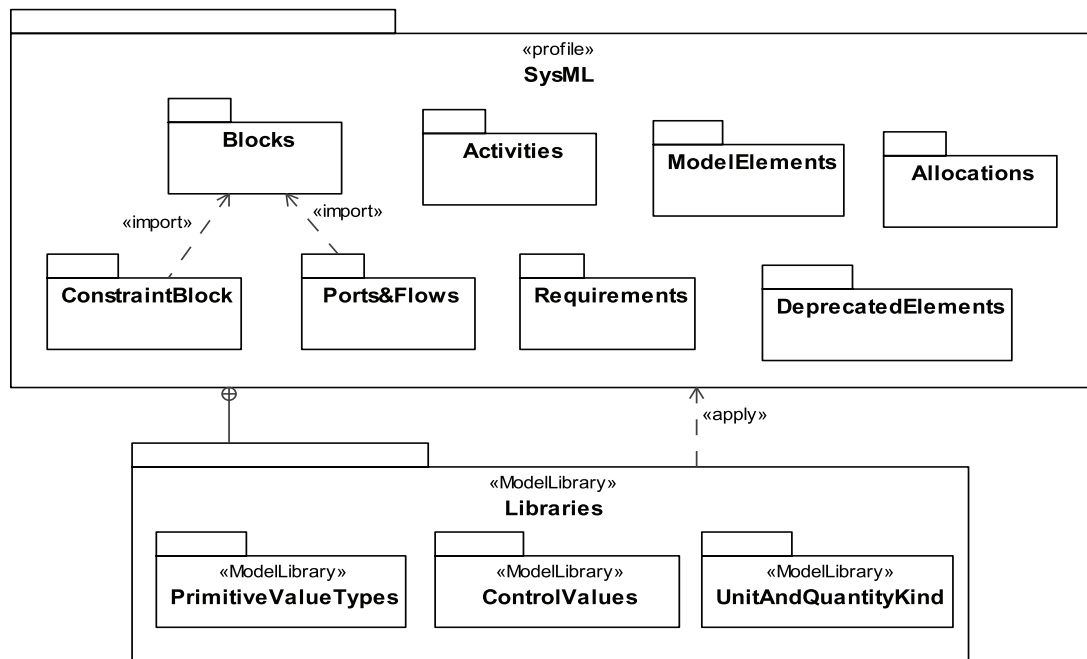


Figure 4.3 - SysML Package Structure

As previously stated, the design approach for SysML is to reuse a subset of UML and create extensions to support the specific concepts needed to satisfy the requirements in the UML for SE RFP. The SysML package structure shown in Figure 4.3 contains a set of packages that correspond to concept areas in SysML that have been extended.

The SysML packages extend UML as follows:

- SysML::Model Elements refactors and extends UML classes.
- SysML::Blocks reuses structured classes from composite structures.
- SysML::ConstraintBlocks extends Blocks to support parametric modeling.
- SysML::Ports and Flows extends UML ports, UML information flows, and SysML Blocks.
- SysML::Activities extends UML activities.
- SysML::Allocations extends UML dependencies.
- SysML::Requirements extends UML classes and dependencies.
- SysML::DeprecatedElements extends UML ports, UML interfaces, and SysML Item Flows.

Figure 4.4 shows non-normative packages in this International Standard that depend on SysML and UML. Note that the QUDV and ISO-80000 libraries are described in non-normative annexes to this specification.

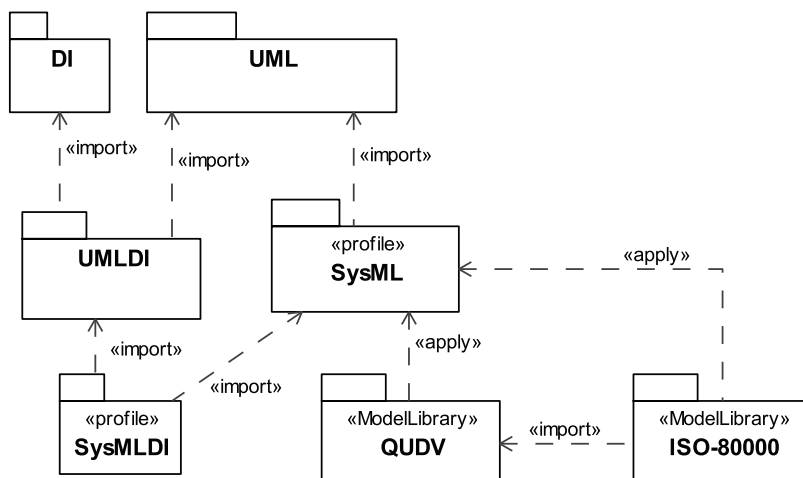


Figure 4.4 - Non-normative Package Structure

4.4 Extension Mechanisms

This International Standard uses the following mechanisms to define the SysML extensions:

- UML stereotypes
- UML diagram extensions
- Model libraries

SysML stereotypes define new modeling constructs by extending existing UML 2 constructs with new properties and constraints. SysML diagram extensions define new diagram notations that supplement diagram notations reused from UML 2. SysML model libraries describe specialized model elements that are available for reuse. Additional non-normative extensions are included in Annex E “Non-normative Extensions.”

The SysML user model is created by instantiating its metamodel and applying the stereotypes specified in the SysML profile, and optionally referencing or subclassing the model elements in the SysML model library. Clause 17, “Profiles & Model Libraries” describes how profiles and model libraries are applied and how they can be used to further extend SysML.

4.5 SysML Diagrams

The SysML diagram taxonomy is shown in Figure A.1 in Annex A. The concrete syntax (notation) for the diagrams along with the corresponding specification of the UML extensions is described in Parts II - IV. The Diagrams Annex (Annex A) describes generalized features of diagrams, such as their frames and headings. A model of SysML diagrams to support interchange is in SysML Diagram Interchange Annex (Annex B).

5 Conformance

5.1 Overview

Conformance with SysML requires that the subset of UML required for SysML is implemented, and that the SysML extensions to this subset are implemented. SysML has three types of conformance, listed in 5.2, which shall all be supported to fully conform to SysML. Conformance does not include `DeprecatedElements`.

5.2 Conformance Types

An implementation of SysML shall comply with both the subset of UML4SysML and the SysML extensions. The types of SysML conformance extend corresponding types in UML as follows:

- *Abstract syntax conformance.* A tool demonstrating abstract syntax conformance provides a user interface and/or API that enables instances of concrete SysML stereotypes (which are applications of stereotypes to instances of UML metaclasses) and model library elements to be created, read, updated, and deleted. The tool shall also provide a way to validate the well-formedness of models that corresponds to the constraints defined in SysML.
- *Concrete syntax conformance.* A tool demonstrating concrete syntax conformance provides a user interface and/or API that enables instances of SysML notation to be created, read, updated, and deleted. This includes conformance to the notation defined in the “Diagram Elements” tables and diagrams extension sub clauses in each clause of this International Standard. Note that a conforming tool may provide the ability to create, read, update, and delete additional diagrams and notational elements that are not defined in SysML.
- *Model interchange conformance.* A tool demonstrating model interchange conformance can import and export conformant XMI for all valid SysML models, including models with profiles defined and/or applied. Model interchange conformance implies abstract syntax conformance. See more information in Annex G.

6 Language Formalism

6.1 Levels of Formalism

SysML is specified using a combination of UML modeling techniques and precise natural language to balance rigor and understandability. Use of more formal constraints and semantics may be applied in future versions to further increase the precision of the language.

6.2 Clause Structure

The clauses are organized according to the SysML packages as described in the language architecture and selected reusable portions of UML 2 packages. This sub clause provides information about how each clause is organized.

6.2.1 Overview

This sub clause provides an overview of the SysML modeling constructs defined in the subject package, which are usually associated with one or more SysML diagram types.

6.2.2 Diagram Elements

This sub clause provides tables that summarize the concrete syntax (notation) and abstract syntax references for the graphic nodes and paths associated with the relevant diagram types. The diagram elements tables are intended to include all of the diagrammatic constructs used in SysML. However, they do not represent all the different combinations in which they can be used. The reader should refer to the usage examples in the clauses and the sample problem (Annex D) for typical usages of the concrete syntax. General diagram information on the use of diagram frames and headings can be found in Annex A.

The diagram elements tables and the additional usage examples fill an important role in defining the scope of SysML. As described in Clause 4, “Language Architecture,” SysML imports many entire packages from the UML metamodel, which it then reuses and extends. Only a subset of the entire UML metamodel, however, is required to support the notations included in SysML.

Unless a type of diagram element is shown in some form in one of the SysML diagram elements tables, or in a usage example in one of the normative SysML clauses, it is not considered to be part of the subset of UML included within SysML, even if the UML metamodel packages support additional constructs. For example, SysML imports the entire Dependencies package from UML, but it includes diagram elements for only a subset of the dependency types defined in this package.

6.2.3 UML Extensions

This sub clause specifies the SysML extensions to UML in terms of diagram extensions and semantic extensions. Diagram extensions are included when the concrete syntax uses notation other than the standard stereotype notation as defined in the Profiles & Model Libraries clause. Semantic extensions consist of stereotype and model library extensions. Stereotype extensions always include the abstract syntax that identifies which metaclasses a stereotype extends. Each stereotype includes a general description with a definition and semantics, along with stereotype properties (attributes), and constraints. Each constraint consists of a textual description and may be followed by a formal constraint expressed in Object Constraint Language (OCL). If there is any ambiguity between the two, the OCL statement of the constraint takes precedence. The model libraries are defined as subclasses of existing metaclasses.

6.2.4 Usage Examples

This sub clause shows how the SysML modeling constructs can be applied to solve systems engineering problems and is intended to reuse and/or elaborate the sample problem in Annex D.

6.3 Conventions and Typography

In the description of SysML, the following conventions have been used:

- When referring to stereotypes, metaclasses, metaassociations, metaattributes, etc. in the text, the exact names as they appear in the model are used.
- No visibilities are presented in the diagrams, since all elements are public.
- If a sub clause is not applicable, it is not included, except for the top-level sub clauses outlined in “Clause Structure” on page 17.
- Stereotype, metaclass, and metaassociation names: initial embedded capitals are used (e.g., “ModelElement,” “ElementReference”).
- Boolean metaattribute names: always start with “is” (e.g., “isComposite”).
- Enumeration types: always end with “Kind” (e.g., “DependencyKind”).

STRUCTURAL CONSTRUCTS

7 Model Elements

7.1 Overview

The ModelElements package of SysML defines general-purpose constructs that may be shown on multiple SysML diagram types. These include package, model, various types of dependencies (e.g., import, access, refine, realization), constraints, and comments. The package diagram defined in this clause is used to organize the model by partitioning model elements into packageable elements and establishing dependencies between the packages and/or model elements within the package. The package defines a namespace for the packageable elements. Model elements from one package can be imported and/or accessed by another package. This organizational principle is intended to help establish unique naming of the model elements and avoid overloading a particular model element name. Packages can also be shown on other diagrams such as the block definition diagram, requirement diagram, and behavior diagrams.

Constraints are used to capture simple constraints associated with one or more model elements and can be represented on several SysML diagrams. The constraint can represent a logical constraint such as an XOR, a condition on a decision branch, or a mathematical expression. The constraint has been significantly enhanced in SysML as specified in Clause 10, “Constraint Blocks” to enable it to be reused and parameterized to support engineering analysis.

Comments can be associated with any model element and are quite useful as an informal means of documenting the model. SysML has introduced an extension to a comment called rationale to facilitate the system modeler in capturing decisions. The rationale may be attached to any entity, such as a system element (block), or to any relationship, such as the satisfy relationship between a design element and a requirement. In the latter case, it may be used to capture the basis for the design decision and may reference an analysis report or trade study for further elaboration of the decision. In addition, SysML includes an extension of a comment to reflect a problem or issue that can be attached to any other model element.

7.1.1 View and Viewpoint

The concepts of viewpoint and view are articulated in ISO-42010 (formerly IEEE-1471). SysML viewpoint and view constructs are consistent with the ISO-42010 standard. Typical examples may include an operational, manufacturing, or security viewpoint and view.

Systems engineers use SysML to make models of systems-the result is the system model, which is what we mean most of the time when we speak of “the model.” Along with that model, systems engineers may also use SysML to make a model of the information to be presented to the stakeholders to address their concerns. The result is the viewpoint and view model, which helps systems engineers assure that stakeholders get the understanding they need from the system model.

The viewpoint and view model can also be thought of as a description model, which augments a system model. A viewpoint and view model exposes elements of one or more system models. In particular, a viewpoint is a specification of rules for constructing a view to address a set of stakeholder concerns. The view is intended to represent the system from this viewpoint. This enables stakeholders to specify aspects of the system model that are important to them from their viewpoint, and then represent those aspects of the system in a specific view.

The viewpoint describes the point of view of a set of stakeholders by framing the concerns of the stakeholders along with the method for producing a view that addresses those concerns. The method describes the expectation of what stakeholder(s) wish to see exposed from the model, how the stakeholder wishes the information to be structured and presented, and in what kind of artifact the stakeholder wants to consume the information. In other words, the method is the set of rules that describe how the view should express the information from the model to address the stakeholder concerns. The method can be specified as a process and/or a set of constraints for producing a view, which may include rules or instructions for analyzing or verifying the view content.

The view is the modeling element that represents the artifact that is presented to the stakeholder. A view conforms to only one viewpoint to ensure that only one method is applied to the view. The view shall be related to the model that contains the information and the method that produces the view. The view is used by a rendering application to generate the artifact, such as a document.

In summary, the viewpoint description specifies the following:

1. What kind of information the view should contain.
2. How the information should be expressed, i.e., what modeling language is required for the model that will appear in the view. (Note: This is not to be confused with the language used for specifying the viewpoint method).
3. The presentation format that specifies how the information should be presented in an artifact, e.g. specifying that data values should be plotted on a graph or a particular tabular style, or that both English and Spanish text should be provided, or that photographs be shown in color with minimum dimensions of 100 millimeters square.
4. The file format of the artifacts that are generated from the view (e.g., set of slides in ppt, a PDF, a Word document, a web viewable format, ...).

It is important to understand that while the view is a SysML construct that exists within a SysML model, artifacts generated from views potentially live outside of the modeling environment as the means to satisfy stakeholder concerns. An artifact such as a movie or a PDF document is not directly incorporated in a SysML model, while the view which represents the artifact does reside in the model as a specification of that artifact. The relationship between the viewpoint and view model and the corresponding artifact is similar to the relationship between the system model and the system that is the subject of the model.

7.2 Diagram Elements

Many of the diagram elements defined in this clause, specifically comments, constraints, problem, rationale, and dependencies, including the dependency subtypes Conform, Realization, and Refine, may be shown on all SysML diagram types, in addition to the diagram elements that are specific to each diagram type.

Table 7.1 - Graphical nodes defined by ModelElements package

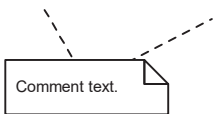
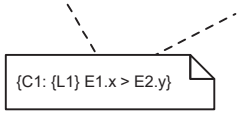
Element Name	Concrete Syntax Example	Abstract Syntax Reference
Comment		UML4SysML::Comment
ConstraintNote		UML4SysML::Constraint

Table 7.1 - Graphical nodes defined by ModelElements package

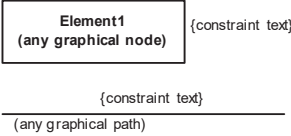
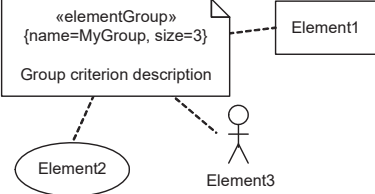
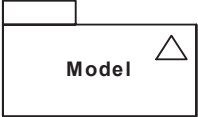
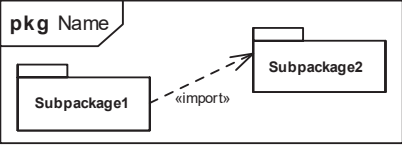
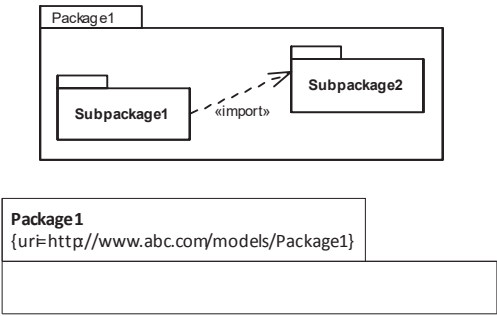
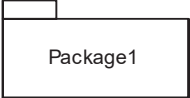
Element Name	Concrete Syntax Example	Abstract Syntax Reference
ConstraintTextualNote		UML4SysML::Constraint
ElementGroup		SysML::ModelElements::Element-Group
Model		UML4SysML::Model
PackageDiagram		UML4SysML::Package
PackageWith NameInTab		UML4SysML::Package
PackageWith NameInside		UML4SysML::Package

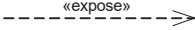
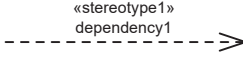
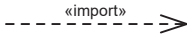
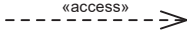
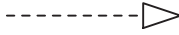
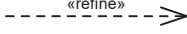
Table 7.1 - Graphical nodes defined by ModelElements package

Element Name	Concrete Syntax Example	Abstract Syntax Reference
Problem		SysML::ModelElements::Problem
Rationale		SysML::ModelElements::Rationale
Stakeholder		SysML::ModelElements::Stakeholder
View		SysML::ModelElements::View
Viewpoint		SysML::ModelElements::Viewpoint

Table 7.2 - Graphical paths defined by ModelElements package

Element Name	Concrete Syntax Example	Abstract Syntax Reference
Conform		SysML::ModelElements::Conform

Table 7.2 - Graphical paths defined by ModelElements package

Element Name	Concrete Syntax Example	Abstract Syntax Reference
Expose		SysML::ModelElements::Expose
Dependency		UML4SysML::Dependency
PublicPackageImport		UML4SysML::PackageImport with visibility = public
PrivatePackageImport		UML4SysML::PackageImport with visibility = private
PackageContainment		UML4SysML::Package:: ownedElement
Realization		UML4SysML::Realization
Refine		UML4SysML::Refine

7.3 UML Extensions

7.3.1 Diagram Extensions

7.3.1.1 UML Diagram Elements not Included in SysML

The notation for a “merge” dependency between packages, using a «merge» keyword on a dashed-line arrow, is not included in SysML. UML uses package merge in the definition of its own metamodel, which SysML builds on, but SysML does not support this capability for user-level models.

NOTE: Combining packages that have the same named elements, resulting in merged definitions of the same names, could cause confusion in user models and adds no inherent modeling capability, and so has been left out of SysML.

7.3.2 Stereotypes

Package ModelElements

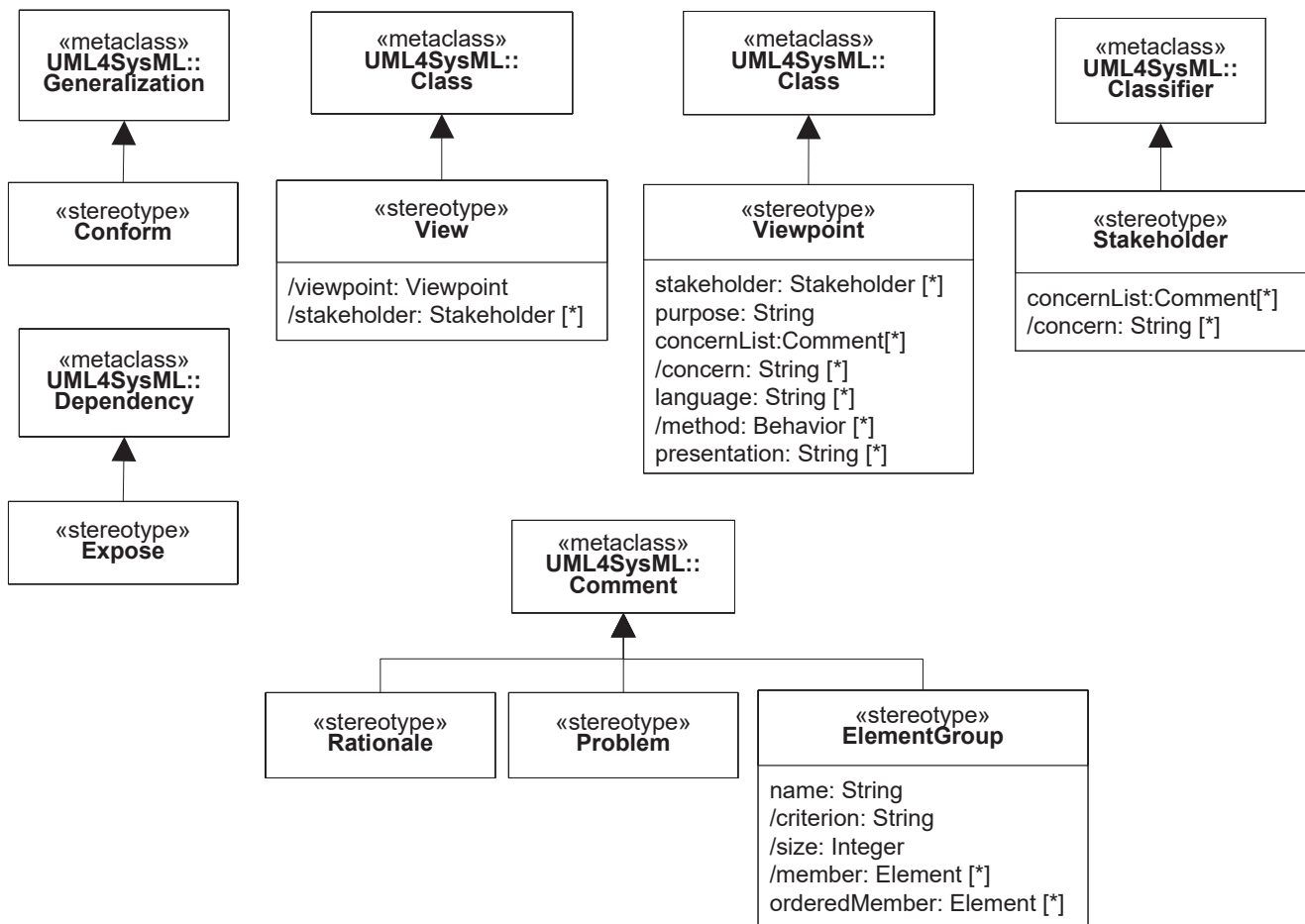


Figure 7.1 - Stereotypes defined in package ModelElements

7.3.2.1 Conform

Description

A Conform relationship is a generalization between a view and a viewpoint. The view conforms to the specified rules and conventions detailed in the viewpoint. When this is done, the view is said to conform to the viewpoint. Conform extends UML generalization.

Constraints

[1] The general classifier shall be an element stereotyped by Viewpoint.

[2] The specific classifier shall be an element that is stereotyped by View.

7.3.2.2 ElementGroup

Description

The ElementGroup stereotype provides a lightweight mechanism for grouping various and possibly heterogeneous model elements by extending the capability of comments to refer to multiple annotated elements. For example, it can group elements that are associated with a particular release of the model, have a certain risk level, or are associated with a legacy design. The semantics of ElementGroup is modeler-defined. In particular, the body text is not restricted. It can describe the grouped elements as well as elements or values related to the grouped elements.

Element groups are named using the name property. The criterion for membership in an element group is specified by the body of the comment the stereotype is applied to. By grouping elements, the modeler asserts that the criterion of the group applies to the member. Optionally, members of an element group can be ordered using its orderedMember property.

ElementGroups appear in diagrams as comments, and properties of the stereotype appear in the notation for stereotype properties. Grouped elements are the annotated elements of the comment to which the stereotype is applied. This has several implications:

- Element groups do not own their elements and thus an element can participate in an unlimited number of groups.
- The elements in a group are identified by the modeler, as opposed to being the result of a query, as in views.
- Element groups can be members of other element groups, but this does not imply that members of the first are members of the second.

Elements related to the grouped elements are not included in the group, even though the body text can address them. In particular, element groups annotating deeply nested properties or properties with bindings are grouping only the properties, rather than their nesting or their bound properties.

Grouped elements are also limited to elements of models, rather than instances of values of those model elements. In particular, element groups annotating blocks or properties are not grouping the instances of the blocks or the values of the properties. However, since the semantics of ElementGroup is left to the modeler, the body text can refer to related elements outside the group, such as instances and values of the grouped elements, or to bound properties. The modeler is then responsible for writing body text that explains the implications for the related elements. For instance:

- A group with the criterion: “Authored by John” could annotate any model element added in the model by John. This body text does not address any related elements. For example, if the annotated element is a property bound to another property, the group would not imply authorship of the second property.
- A group with the criterion: “Instances are manufactured in a foreign country” could annotate Blocks to indicate that any instances of those Blocks are produced in a foreign country. This body text does not address the Block itself, which is not necessarily “manufactured” in a foreign country.
- A group with criterion: “Values are manufactured in a foreign country” could annotate properties, including part properties, to indicate the values of the property are produced in a foreign country. This body text does not address the property itself, which is not necessarily “manufactured” in a foreign country. Since the text is about values of the property, it is also about values of other properties that might be bound to the annotated property, because the values of bound properties are the same.

Attributes

- name: String
Name of the element group

- `/criterion[0..1]: String`
Specifies the rationale for being member of the group. Adding an element to the group asserts that the criterion applies to this element.
Derived from `Comment::body`.
- `/size: Integer`
Number of members in the group. Derived.
- `/member: Element[0..*]`
Set specifying the members of the group.
Derived from `Comment::annotatedElement`.
- `orderedMember: Element[0..*] {ordered, subsets member}`
Organize member according to an arbitrary order. Optional.

Operations

- [1] The query `criterion()` returns the text describing the criterion defining the group.
- ```
criterion(): String[0..1] {query}
body: self.base_Comment.body
```
- [2] The query `size()` returns the number of elements which are members of the group.
- ```
size(): Integer[1..1] {query}
body: self.base_Comment.annotatedElement->size()
```
- [3] The query `member()` returns the set of all the members of the group.
- ```
member(): Set(Element)[0..*] {query}
body: self.base_Comment.annotatedElement
```
- [4] The query `AllGroups()` returns the set of all the groups an element is member of.
- ```
allGroups(e:Element): Set(ElementGroup)[0..*] {query, static}
body: ElementGroup.allInstances()->select(member->includes(e))
```

7.3.2.3 Expose

Description

The expose relationship relates a view to one or more model elements. Each model element is an access point to initiate the query. The view and the model elements related to the view are passed to the constructor when it is invoked. The method describes how the exposed elements are navigated to extract the desired information.

Constraints

- [1] The client shall be an element stereotyped by View.

7.3.2.4 Problem

Description

A Problem documents a deficiency, limitation, or failure of one or more model elements to satisfy a requirement or need, or other undesired outcome. It may be used to capture problems identified during analysis, design, verification, or manufacture and associate the problem with the relevant model elements. Problem is a stereotype of comment and may be attached to any other model element in the same manner as a comment.