Date: February 2024

Standards
Development
Organization®

Specification Common Elements (SCE)

Version 1.0 Beta 2

OMG Document Number: dtc/2024-04-07

Normative reference: https://www.omg.org/spec/SCE/1.0/Beta2

Machine readable file(s): https://www.omg.org/SCE/20240210

Normative:

https://www.omg.org/spec/SCE/20240210/SCE.xsd
https://www.omg.org/spec/SCE/20240210/SCEDI.xsd
https://www.omg.org/spec/SCE/20240210/DI.xsd
https://www.omg.org/spec/SCE/20240210/DC.xsd
https://www.omg.org/spec/SCE/20240210/SCE-Library.xml
https://www.omg.org/spec/SCE/20240210/SCE.xmi
https://www.omg.org/spec/SCE/20240210/SCEDI.xmi
https://www.omg.org/spec/SCE/20240210/SCE-Library.xmi

Informative:

https://www.omg.org/spec/SCE/20240210/SCE.mdzip
https://www.omg.org/spec/SCE/20240210/SCE-diagrams.zip

Commented [SW1]: This convenience document provides the changes (deletions and additions) to the SCE specification based on the resolutions for issues raised for the SCE FTF. For this document, the resolutions for the issues that have been through a set of SCE FTF Ballots have been applied.
A comment is attached to each change in the document. The comment identifies the type of change (e.g., a figure update) and the raised Issue and its resolution sub-task. Thus, the issues will be identified as so (e.g.): SCE-1/SCE-50.
By searching through the document for a particular issue (e.g., SCE-1), you can find all the changes to the specification based on the resolution for that issue.

Commented [SW2]: This document provides the specification changes for the resolution of the following issues:
Issue SCE-9/SCE-65
Issue SCE-12/SCE-68
Issue SCE-14/SCE-72
Issue SCE-27/SCE-30
Issue SCE-35/SCE-81
Issue SCE-36/SCE-76
Issue SCE-40/SCE-41
Issue SCE-47/SCE-79
Issue SCE-48/SCE-82
Issue SCE-53/SCE-84
Issue SCE-62/SCE-80
Issue SCE-63/SCE-64
Issue SCE-67/SCE-78
Issue SCE-85/SCE-86
Issue SCE-90/SCE-95
Issue SCE-91/SCE-93
Issue SCE-92/SCE-94
Issue SCE-96/SCE-97
Issue SCE-103/SCE-104
Issue SCE-105/SCE-106
Issue SCE-107/SCE-108
Issue SCE-109/SCE-110
Issue SCE-111/SCE-112
Issue SCE-113/SCE-114
Issue SCE-115/SCE-116
Issue SCE-117/SCE-118
Issue SCE-119/SCE-120
Issue SCE-121/SCE-122

Commented [SW3]: This text was updated for the resolution of Issue SCE-91/SCE-93. Updating copyright information.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 9C Medway Road PMB 274, Milford, MA, 01757 U.S.A.

TRADEMARKS

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Specifications, Report a Bug/Issue.

Table of Contents

Commented [SW4]: The content tables were updated to reflect changes in structure, figures, and tables.

Preface

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at https://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from the OMG website at:

https://www.omg.org/spec

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult https://www.iso.org

Issues

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page https://www.omg.org, under Documents, Report a Bug/Issue.

1 Scope

The primary goal of **SCE** is to provide a set of structural elements that are common to other OMG specifications. The ~~proposed~~ specification~~s, **BKPMN, PPMN, and**~~ **SDMN**~~, are~~ is structured to be dependent on the elements defined in **SCE**. Other BMI and HDTF specifications may also utilize the elements of **SCE** as they are updated in the future.

Commented [SW5]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

2 Conformance

SCE is not an independent specification that is implemented by itself. It is used by other specifications to provide generic capabilities that can be used by those other specifications. At the time of this writing~~, the BPM+ Knowledge Package Model and Notation (**BKPMN**),~~ the Shared Data Model and Notation (**SDMN**)~~, and the Pedigree and Provenance Model and Notation (**PPMN**)~~ specification~~s are~~ is dependent on **SCE**.

Commented [SW6]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

Software that claims compliance or conformance to any specification that is dependent of **SCE** if and only if the software fully matches the applicable compliance points as stated in the dependent specification and this specification. Software developed only partially matching the applicable compliance points can claim only that the software was based on this specification but cannot claim compliance or conformance with this specification.

3 References

3.1 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, IETF RFC 2119, March 1997 http://www.ietf.org/rfc/rfc2119.txt
- [DD] Diagram Definition (DD™): https://www.omg.org/spec/DD/™~~)~~
- [MOF] Meta Object Facility (MOF™): https://www.omg.org/spec/MOF/ ~~https://www.omg.org/spec/MOF/~~
- [UML] Unified Modeling Language ™ (UML®): http://www.omg.org/spec/UML
- [XMI] XML Metadata Interchange (XMI®) http://www.omg.org/spec/XMI

Commented [SW7]: This text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues.

3.2 Non-normative References

The following normative documents contain provisions which, through reference in this text, constitute exemplars or influencers of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- [BPMN] ~~OMG~~ Business Process and Model Notation (BPMN™): https://www.omg.org/bpmn/
- [CMMN] ~~OMG~~ Case Management Model and Model Notation (CMMN™): https://www.omg.org/spec/CMMN/
- [DMN] ~~OMG~~ Decision Model and Model Notation (DMN™): https://www.omg.org/spec/DMN/ ~~https://www.omg.org/spec/DMN/~~
- [MDMI] ~~OMG~~ Model Driven Message Interoperability (MDMI), Version 1.0: https://www.omg.org/spec/MDMI/ ~~https://www.omg.org/spec/MDMI/~~
- [SysML] ~~OMG~~ Systems Modeling Language (SysML®): http://www.omg.org/spec/SysML/

Commented [SW8]: This text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues.

4 Terms and Definitions

The table below presents a glossary for this specification:

Table 1. Glossary

Term	Definition
Case	A **CMMN** element that is a proceeding that involves actions taken regarding a subject in a particular situation to achieve a desired outcome.
DataItem	A **SDMN** DataItem represents a common definition and structure for the data handling elements of the other BPM+ models.
DataState	DataItemscan optionally reference a DataState element, which is the state of the data contained in the DataItem. The definition of these DataStates, e.g., possible values and any specific semantic are out of scope of this specification. Therefore, **SDMN** adopters can use the DataState element and the **SDMN** extensibility capabilities to define their DataStates.
Decision	A **DMN** element that is the act of determining an output value (the chosen option), from a number of input values, using logic defining how the output is determined from the inputs.
Process	A **BPMN** element that describes a sequence or flow of Activities in an organization with the objective of carrying out work. The ProcessRef element provides a link to a Process in a **BPMN** document.

5 Symbols

There are no symbols defined in this specification.

6 Additional Information

6.1 Conventions

The section introduces the conventions used in this document. This includes (text) notational conventions and notations for schema components. Also included are designated namespace definitions.

6.2 Typographical and Linguistic Conventions and Style

This document incorporates the following conventions:

- The keywords “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “MUST NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document are to be interpreted as described in RFC-2119.
- A **term** is a word or phrase that has a special meaning. When a term is defined, the term name is highlighted in **bold** typeface.
- A reference to another definition, section, or specification is highlighted with underlined typeface and provides a link to the relevant location in this specification.
- A reference to a graphical element is highlighted with a bold, capitalized word (e.g., **Process**).
- A reference to a non-graphical element or **SCE** concept is highlighted by being italicized and (e.g., *Documentation*).
- A reference to an attribute or model association will be presented with the `Courier New` font (e.g., `Expression`).

- Non-normative examples are set off in boxes and accompanied by a brief explanation.
- XML and pseudo code is highlighted with `Courier New` typeface. Different font colors MAY be used to highlight the different components of the XML code.
- The cardinality of any content part is specified using the following operators:
  - <none> — exactly once
  - [0..1] — 0 or 1
  - [0..*] — 0 or more
  - [1..*] — 1 or more
- Attributes separated by | and grouped within { and } — alternative values
  - <value> — default value
  - <type> — the type of the attribute

6.3 Display of Metamodel Diagrams

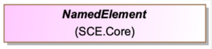
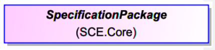
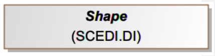
The metamodel presented in these sections utilizes the patterns and mechanisms that are used for the current BPM+ specifications. OMG specifications rarely display the entire metamodel of a technical specification in a single diagram. The entire metamodel would be very large, complicated, and hard to follow. Typically, a specification will present sub-sets of the overall metamodel as they apply to specific topics. For example, in the **BPMN** specifications there are metamodel diagrams that show the elements relating to activities or data elements. This document will follow that pattern and present sub-sets of a larger metamodel.

The metamodel diagrams are Unified Modeling Language (UML) structure diagrams. In addition to the metamodel, OMG specifications provide XML schemas which map to the metamodels. In general, it is through XML documents that BPM+ models are stored and exchanged.

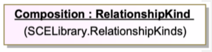
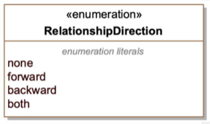
Further, some of the metamodel elements are references to elements from other specifications. To clarify the owner of the metamodel element, there is a parenthesized text that identifies the model owner of that element. In addition, colors are used to support the text identification of the owner-language of that element. The colors are used as an aid to distinguish the languages but does not represent a normative aspect of the metamodels nor do they add any semantic information about the metamodels.

The table below presents examples of elements used throughout the metamodel diagrams within this specification:

Table 2. SCE Metamodel Color-Coding

Element	Description	Example Color
SCE General Class	These elements elements include the owner of the language (SCE) in parenthases below the element name and these elements are color-coded violet to distinguish **SCE** classes from related **BPM+** specification classes (e.g., **SDMN** ~~or **BKPMN**~~) (see figure to the right).	
SCE General Class (focus of diagram)	These elements have the same naming and color, but the border line color is dark blue instead of light brown (see figure to the right). They are highlighted as the focus of the particular metamodel diagram. This is an informative depiction that does not add any semantic information about the particular metamodel diagram.	
External Class	Classes from specifications that are not specifically part of the **BPM+** stack of standards can be included in metamodel diagrams and display the owner of the language in parenthases below the element name and these elements are color-coded light-gray. (see figure to the right).	

Commented [SW9]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

SCE Class Instance	These elements elements include the owner of the language (SCE) in parenthases below the element name and these elements are color-coded light-violet to identify **SCE** class instances from the **SCE** Library (see figure to the right).	 Composition : RelationshipKind (SCELibrary.RelationshipKinds)
Enumerations	(see figure to the right).	 «enumeration» RelationshipDirection enumeration literals none forward backward both

6.4 Use of Text, Color, Size, and Lines in a Diagram

- Diagram elements MAY have labels (e.g., its name and/or other attributes) placed inside the shape, or above or below the shape, in any direction or location, depending on the preference of the modeler or modeling tool vendor.
- The fills that are used for the graphical elements MAY be white or clear.
  - The notation MAY be extended to use other fill colors to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute).
- Diagram elements and markers MAY be of any size that suits the purposes of the modeler or modeling tool.
- The lines that are used to draw the graphical elements MAY be black.
  - The notation MAY be extended to use other line colors to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute).
  - The notation MAY be extended to use other line styles to suit the purpose of the modeler or tool (e.g., to highlight the value of an object attribute) with the condition that the line style MUST NOT conflict with any current defined line style of the diagram.

Note: The requirements specified in this section are specifically focused on `DiagramArtifacts` *(see below). Any modeling specification that is dependent on* ***SCE*** *will define its own diagram requirements, which may override the items listed here.*

6.5 Abbreviations

The table below presents a list of acronyms, and their defintion, that are used in this specification:

Table 3. Acronyms

Acronym	Definition
~~BKPMN~~	~~BPM+ Knowledge Package Model and Notation~~
BPM+	Business Process Management Plus
BPMN	Business Process Model and Notation
CMMN	Case Management Model and Notation
DC	Diagram Commons
DD	Diagram Definition
DI	Diagram Interchange
DMN	Decision Model and Notation
MOF	Meta Object Facility
OMG	Object Management Group
~~PPMN~~	~~Provenance and Pedigree Model and Notation~~
RFC	Remote Function Call
SCE	Specification Common Elements
SCEDI	Specification Common Elements Diagram Interchange
SDMN	Shared Data Model and Notation

Commented [SW10]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

Commented [SAW11]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

SysML	Systems Modeling Language
URI	Uniform Resource Identifier
XMI	XML Metadata Interchange
XML	Extensible Markup Language

6.6 Structure of this Document

This document provides a brief introduction to **SCE** and its purpose (see the section entitled "Overview"). ~~**Error! Reference source not found.**~~"). ~~Overview").~~ The introduction is followed by normative clauses that define the elements of the specification and their properties and associations (see the sections entitled "SCE Metamodel" (Clause 8); "SCE Library" (Clause 9); and "SCE Diagram Interchange" (Clause 11)).

6.7 Acknowledgements

The following companies submitted version 1.0 of this specification~~Submitting Organizations (RFP Process)~~

- Auxilium Technology Group, LLC
- BPM Advantage Consulting, Inc.

The following companies supported this specification~~Supporting Organizations (RFP Process)~~

~~The following organizations support this specification but are not formal submitters:~~

- Adaptive
- agnos.ai UK Ltd
- Airbus Group
- BookZurman, Inc.
- Camunda Services GmbH
- Department of Veterans Affairs
- FICO
- Mayo Clinic
- MDIX, Inc.
- Red Hat
- Thematix Partners, LLC
- Trisotech
- University of Utah
- XZYOS, LLC

Special Acknowledgements

The following persons were members of the core teams that contributed to the content of this specification (in alphabetical order): James D. Baker, Maciej Barelkowski, Thomas Beale,~~:~~ Claude Baudoin, John Butler, Keith Butler, Lloyd Dugga~~e~~n~~Duggen~~, Denis Gagne, Eder Ignatowicz, Peter Haug, Elisa Kendall, Matteo Mortari, Falko Menge, Sean Muir, Robert Lario, Kenneth~~Ken~~ Lord, Simon Ringuette, Peter Rivett, Michael Sauvage, Keith Salzman, Jane Shellum, Davide Sottara, and Stephen A. White.

7 Overview

Commented [SW12]: This text was changed for Issue SCE-11/SCE-51

Commented [SW13]: This text updated for the resolution of Issue SCE-92/SCE-94. Update Acknowlegements

The idea for defining a model for Specification Common Elements (**SCE**) occurred during the development of the ~~**BKPMN** and~~ **SDMN** specification (and other possible future specifications)~~s~~. These specifications were developed using patterns seen in OMG Business Modeling and Integration (BMI) Task Force, such as **BPMN** and **DMN**. ~~Both **BKPMN** and~~ **SDMN** shares~~d~~ a common set of ~~8~~ elements and their attributes with these specifications. ~~**PPMN** also shared these elements.~~ Thus, the purpose of **SCE** is to provide a set of structural elements that are common to these and other OMG specifications. ~~**BKPMN, PPMN,** and~~ **SDMN** ~~have~~ has been structured to be dependent on the elements defined in **SCE**. Other BMI and HDTF specifications may also utilize the elements of **SCE** as they are updated in the future.

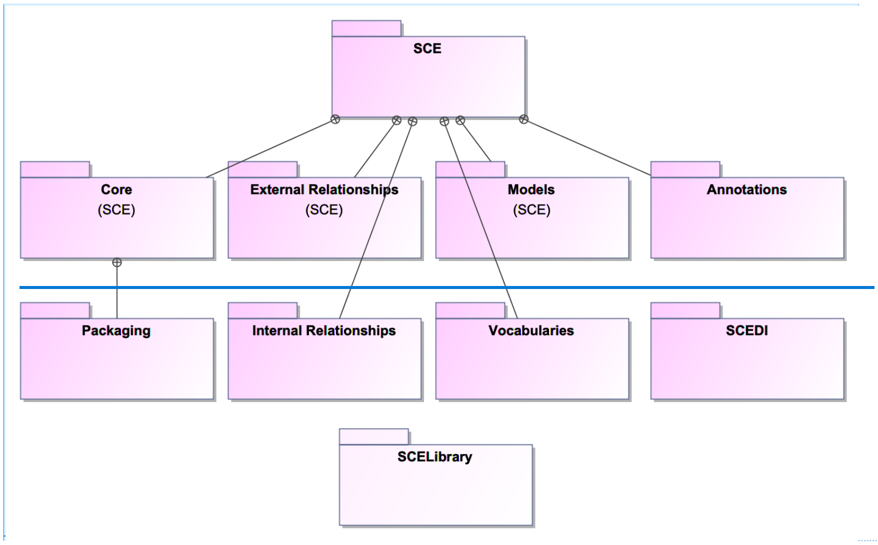
Commented [SW14]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

8 SCE Metamodel

This section defines the semantic elements of **SCE**. The main topics are organized into **SCE** Core Elements, Annotations, External Relationships, Internal Relationships, BPM+ Modeling, and ~~KindSetsKindSetsKindSetsVocabularies~~KindSets.

Commented [SW15]: this text was changed for Issue SCE-7/SCE-8 (Semantic Reference changes)

The following figure shows the organization of the **SCE** metamodel packages.



Commented [SW16]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes). The Vocabularies package was renamed to KindSets

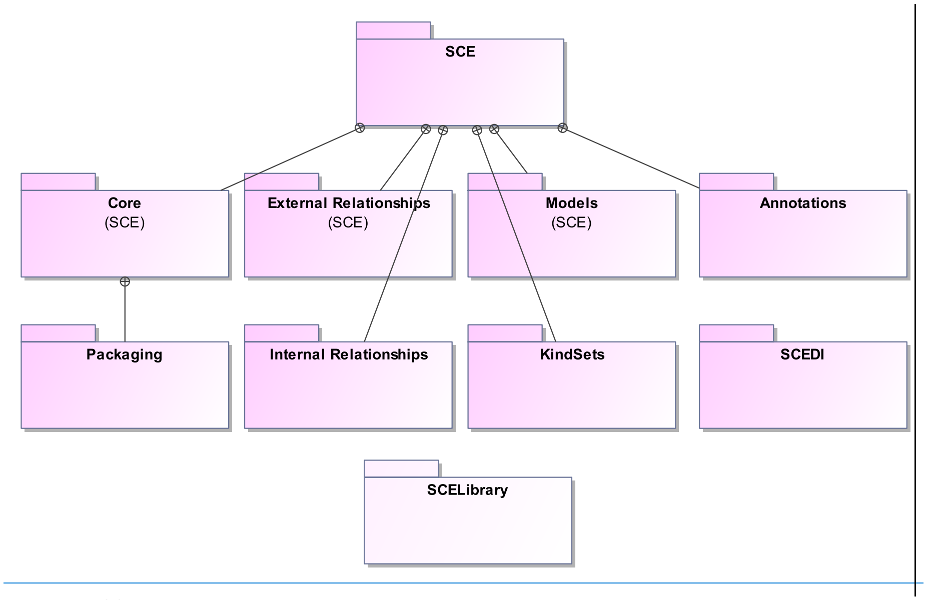


Figure 1 - SCE Packages

8.1 SCE Core Elements

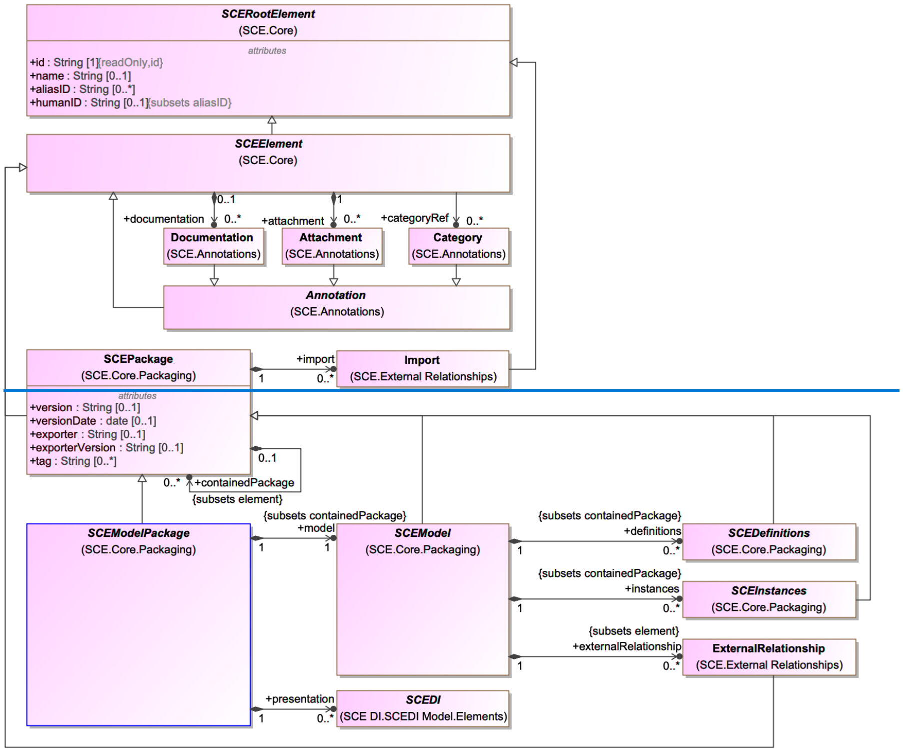
There ~~are~~ is ~~two~~ one core abstract element~~s~~ that make up **SCE** with a few supporting elements. The core element~~s~~ ~~are~~is: ~~*SCERootElement* and *SCEElement*~~*BaseElement*.~~.~~ There are ~~six~~ ~~three~~two elements related to the packaging of **SCE** elements (and downstream languages). These are defined in the sub-section below.

The following figure presents the **SCE** high-level metamodel, which defines the basic infrastructure elements of a **BPM+** model:

Commented [SW17]: This text was updated from the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. merging RootElement and Element.

Commented [SW18]: This text was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting 3 packages: SCEProfile, SCE Definitions, and SCEModel.

Commented [SW19]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.
This involved multiple text find and replaces. All such changes are part of that resolution. The numbers of changes are as follows:
SCEElement to Element 121 times (All of these changes were further changed: Element was replaced by BaseElement for SCE-96/SCE-97 and SCE-121/SCE-122)
SCEDiagram to Diagram 77 times
SCERootElement to RootElement 60 times
SCEPackage to Package 48 times
SCEDiagramElement to DiagramElement 45 times
SCEEdge to Edge 44 times
SCEModel to Model 39 times
SCEShape to Shape 35 times
SCEStyle to Style 33 times
SCEKindSet to KindSet 29 times
SCEInstance to Instances 23 times
SCEDI to Diagrams 4 times



Commented [SW20]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the “SCE” prefix for model elements.

Commented [SW21]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes). The conceptReference attribute was added to SCEElement.

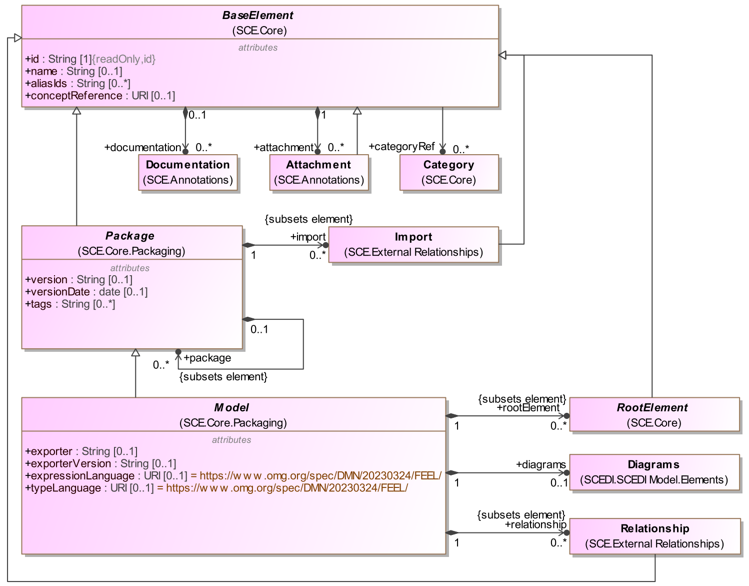


Figure 2 - SCE High-Level Elements

Commented [SW22]: This figure was also updated for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes. RootElement and Element merged and renamed to BaseElement. A new RootElement added.

Commented [SW23]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

Commented [SW24]: This figure was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW25]: This figure was also updated for the resolution of Issue SCE-105/SCE-106. changing tag to tags

Commented [SW26]: Also, this Figure was replaced for Issue SCE-2/SCE-52 (Add expressionLanguage and typeLanguage to the SCEModelPackage Element)

Commented [SW27]: This figure was also updated resolution of Issue SCE-107/SCE-108. Removing Annotation Class.

Commented [SW28]: This figure was also updated for the resolution for Issue SCE-67/SCE-78. Setting expressionLanguage and typeLanguage to optional.

Commented [SW29]: This figure was updated for the resolution of Issue SCE-85/SCE-86. Move exporter and exporterVersion.

Commented [SW30]: This figure was also updated for the resolution for Issue SCE-47/SCE-79. changing capitalization of humanId and aliasId.

Commented [SW31]: This figure was also updated for the resolution of Issue SCE-27/SCE-30. Editorial issues. humanId should have been removed from RootElement.

Commented [SW32]: This figure was also updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

~~8.1.1 SCERootElement~~

~~*SCERootElement* is the abstract super class for most **SCE** elements. Basically, it is the root element of the **SCE** metamodel. All the elements within **SCE**, and any specification that is dependent on **SCE**, will inherit the attributes of *SCERootElement*. It provides the basic attributes for `id` and `name`.~~

~~Generalizations~~

~~The *SCERootElement* element does not inherit any attributes or associations of from another element.~~

~~Properties~~

~~The following table presents the additional attributes and/or associations for *SCERootElement*:~~

~~**Table 4. SCERootElement Attributes and/or Associations**~~

~~**Property/Association**~~	~~**Description**~~
~~**aliasIdsaliasIdsaliasIdsDaliasID** : String [0..*]~~	~~Various optional, alternative identifiers for this *SCERootElement.SCERootElement*Element. Generally, these will be set by tools.., but one of them (the `humanId`), in particular, may be set by the modeler.~~
~~**humanID** : String [0..1]~~	~~An identifier for this element that is set by the modeler. It is the responsibility of the modeler to maintain the uniqueness of this identifier within a model or relative to some other context.~~

Commented [SW33]: This attribute was updated for the resolution of Issue SCE-48/SCE-83. removed humanId and update aliasID

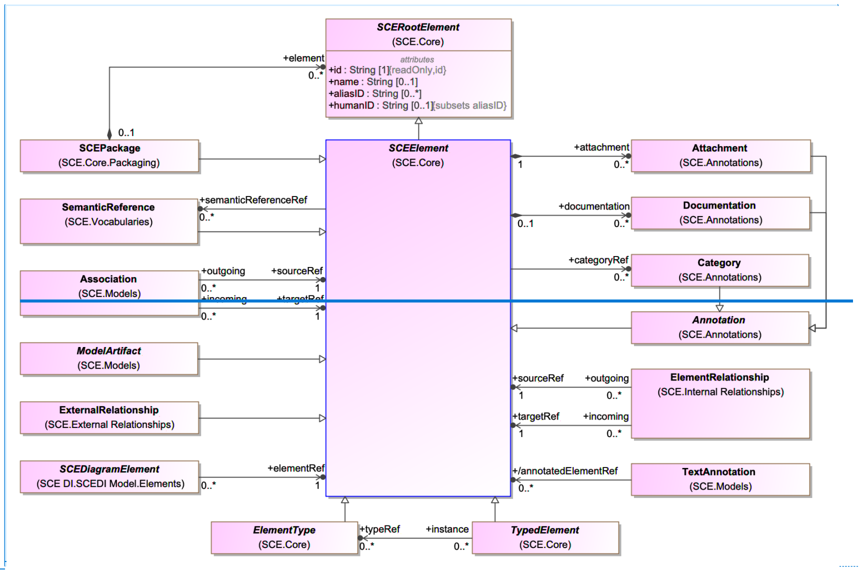
Commented [SW34]: This attribute was removed for the resolution of Issue SCE-48/SCE-83. removed humanId and update aliasID

~~**id** : String [0..1]~~	~~This optional attribute is used to uniquely identify a *SCERootElement*. The `id` is REQUIRED if this element is referenced or intended to be referenced by something else. If the element is not currently referenced and is never intended to be referenced, the `id` MAY be omitted.~~
~~**name** : String [0..1]~~	~~The `name` attribute is a text description or label of the element. In general, the name is optional, but many elements will require a name. The definition of each specialization of *SCERootElement* may require name to be mandatory.mandatoryidentify this requirement.~~

~~8.1.2~~8.1.1~~SCEElement~~BaseElement~~SCEElement~~

~~*SCEElement* extends *SCERootElement* with a set of common associations, such as `documentation`, that are useful for most elements of a modeling language. Most~~ Most of the elements within **SCE**, and any specification that is dependent on **SCE**, will inherit the attributes and associations of *~~SCEElement~~BaseElement*.

The following figure presents the metamodel for *~~SCEElement~~BaseElement:~~.~~*



Commented [SW35]: This attribute was updated for the resolution of Issue SCE-12/SCE-68. make id optional.

Commented [SW36]: This text was updated for the resolution of Issue SCE-13/SCE-60

Commented [SW38]: This Title was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. RootElement and Element were merged and named BaseElement. Note that every change from Element to BaseElement WILL NOT be marked with a comment. There are dozens of such changes.

Commented [SAW39]: This text was updated from the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. merging RootElement and Element.

Commented [SW40]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW41]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes). the conceptReference attributed was added to SCEElement and the association to SemanticReference was removed.

Commented [SW43]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes)

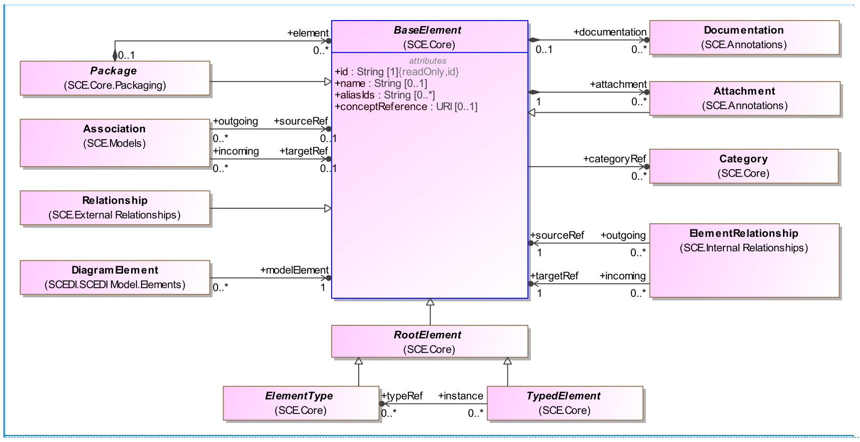


Figure 3 - The ~~SCEElement~~BaseElement Metamodel

Commented [SW42]: This figure was also updated for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes. RootElement and Element merged and renamed to BaseElement. A new RootElement added.

Commented [SW44]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

Commented [SW45]: This figure was updated for the resolution of Issue SCE-27/SCE-30. Editorial issues. humanId should have been removed from RootElement.

Commented [SW46]: This figure was also updated resolution of Issue SCE-107/SCE-108. Removing Annotation Class.

Generalizations

The *~~SCEElement~~BaseElement* element does not inherit~~s~~ the attributes and/or associations of another element~~:~~.

- ~~*SCERootElement* (see the section entitled "SCERootElementSCERootElement" for more information).~~

Properties

The following table presents the additional attributes and/or associations for *~~SCEElement~~BaseElement*:

~~Table 5.~~Table 4. ~~SCEElement~~BaseElement Attributes and/or Associations

Property/Association	Description
aliasIds : String [0..*]	Various optional, alternative identifiers for this *BaseElement*. Generally, these will be set by tools.
id : String [0..1]	This optional attribute is used to uniquely identify a *BaseElement*. The `id` is REQUIRED if this element is referenced or intended to be referenced by something else. If the element is not currently referenced and is never intended to be referenced, the `id` MAY be omitted.
name : String [0..1]	The `name` attribute is a text description or label of the element. In general, the name is optional, but many elements will require a name. The definition of each specialization of *BaseElement* may require name to be mandatory.
attachment : Attachment [0..*]	This association is used to annotate any concrete specialization of *~~SCEElement~~BaseElement* with descriptions and other documentation.
categoryRef : Category [0..*]	This association is used to categorize any concrete specialization of *~~SCEElement~~BaseElement*. A *Category* has user-defined semantics, which can be used for documentation or analysis purposes.

Commented [SAW47]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW48]: These rows were moved from the deleted RootElement to the new BaseElement for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes. RootElement and Element merged and renamed to BaseElement.

Commented [SAW49]: This text was updated from the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. merging RootElement and Element.

conceptReference : URI [0..1]	The specific context of the BPM+ elements may result in different terminology or sub-sets of data representation elements within the normative domain models. To reduce any confusion due to terminology or data representation, the BPM+ models dependent on **SCE** have the capability of linking model elements to the appropriate external sources of truth for their domain (i.e., a `conceptReference`). This property provides the capability of including a ~~`conceptReference`~~`conceptReference`Concept Reference for any ~~*SCEElement*~~*BaseElement*. It is expected that the value of the URI will be persistent.
documentation : Documentation [0..*]	This association is used to annotate any concrete specialization of ~~*SCEElement*~~*BaseElement* with descriptions and other documentation.
~~**semanticReferenceRef** : SemanticReference [0..*]~~	~~A concrete *SCEElement* can reference zero or more *SemanticReference* elements.~~

Commented [SW50]: conceptReference was added for Issue SCE-7/SCE-8 (Semantic Reference changes)

Commented [SAW51]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

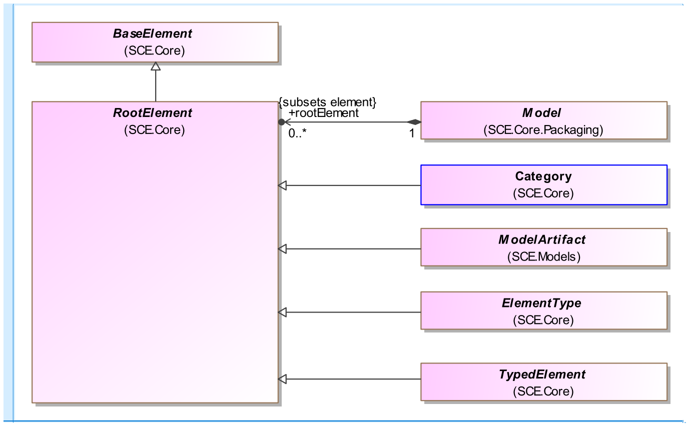
Commented [SW52]: semanticReferenceRef was removed for Issue SCE-7/SCE-8 (Semantic Reference changes)

8.1.2 RootElement

The *RootElement* class is also a marker class for **BPM+** languages to use to include their specific model elements within their models (e.g., subclasses of the **SCE** *Model* class). A **BPM+** language would define its model elements as subclasses of *RootElement* and they would all be a part of the *Model* (see the *RootElement* relationship to *Model* in the figure entitled "The SCE Packaging Elements Metamodel~~The SCE Packaging Elements Metamodel~~", below).

Further, in the XML schema for **SCE**, *RootElement* is also used as a substitution group for all model elements to be included in a *Model*. Thus, a **BPM+** language would not have to specific list all the elements within its version of *Model*. The language model elements would only have to be subclasses of *RootElement* and they would automatically be included in the *Model* and a specific ordering of those elements would not be required.

The following figure presents the metamodel for *RootElement*:



Commented [SW53]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

Figure 4 - The RootElement Metamodel

~~Generalizations~~Generalizations

The *RootElement* element inherits the attributes and/or associations of:

- *BaseElement* (see the section entitled “BaseElement~~BaseElement~~” for more information).

~~Properties~~Properties

The *RootElement* element does not have any additional attributes and/or associations.

Commented [SW54]: This section was added for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes. RootElement and Element merged and renamed to BaseElement. A new RootElement added.

8.1.3 ElementType

A kind of ~~*SCEElement*~~ *BaseElement* (via *RootElement*) that can be a type or specification of a *TypedElement*. This usually is applied to the concrete *TypedElement* that serves as an instance in a runtime model.

An example of a *ElementType* in the context of Provenance and Pedigree would be the entity-type “Thoroughbred Horse” that is used to specific the basic characteristics of thoroughbred horses. The entity “Secretariat” (the horse), which is a *TypedElement*, is, in a sense, an “instance” of the entity-type “Thoroughbred Horse”.

Generalizations

The *ElementType* element inherits the attributes and/or associations of:

- ~~*SCEElement*~~ *RootElement* (see the section entitled “RootElement”~~“SCEElement”~~ for more information).

Further, the ~~*SCEElement*~~ *RootElement* element inherits the attributes and/or associations of:

- ~~*SCERootElement*~~*BaseElement* (see the section entitled “~~SCERootElement~~BaseElement”~~”~~ for more information).

Commented [SW55]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The *ElementType* element does not have any additional attributes and/or associations.

8.1.4 TypedElement

A kind of ~~*SCEElement*~~ *BaseElement* (via RootElement) that has zero or more *ElementTypes,* identified by the `typeRef` attribute. The *ElementType(s)*, if present, provide a specification for the element.

An example of a *TypedElement* in the context of Provenance and Pedigree would be the entity “Secretariat” (the horse) where the entity’s pedigree is documented. The entity is a *TypedElement* since an *ElementType*, such as “Thoroughbred Horse”, can be used to specify the basic characteristics of thoroughbred horses. The specific entity “Secretariat” is, in a sense, an “instance” of the entity-type “Thoroughbred Horse”.

Generalizations

The *TypedElement* element inherits the attributes and/or associations of:

- ~~*SCEElement*~~ *RootElement* (see the section entitled “RootElement” ~~“SCEElement”~~ for more information).

Further, the ~~*SCEElement*~~ *RootElement* element inherits the attributes and/or associations of:

- *~~SCERootElement~~BaseElement* (see the section entitled "~~SCERootElement~~BaseElementBaseElement" for more information).

Commented [SW56]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for *TypedElement*:

~~Table 6.~~Table 5. TypedElement Attributes and/or Associations

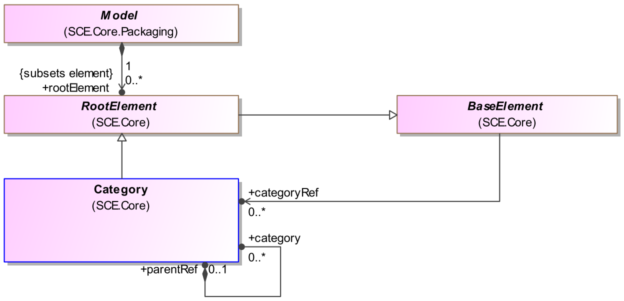
Property/Association	Description
typeRef : ElementType [0..*]	The class(es) that provide(s) a specification, through an *ElementType*, of the *TypedElement*. This usually is applied to the concrete *TypedElement* that serves as an instance in a runtime model.

8.1.5 Category

A *Category*, which has user-defined semantics, can be used for documentation or metadata organizational purposes. For example, recommendations (in the healthcare domain) can be assigned a category of "Lifestyle Modification" with further breakdowns into "Weight Reduction," "Exercise Program," and "Diet Modification" sub-categories.

Commented [SW58]: This section was moved for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. and a new metamodel diagram added.

The following figure presents the metamodel for *Category*:



Commented [SW57]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

Figure 5 - The Category Metamodel

The *Category* element inherits the attributes of *BaseElement via RootElement* and is contained within a *Model* since it is a *RootElement* (see figure above). It is referenced by any *BaseElement*. Thus, any concrete element within a model file, dependent on **SCE**, MAY have zero or more *Categories*. Further, *Categories* may be nested such that one *Category* may contain other *Categories*.

> *Note: The structure of Category in **SCE** is different than the structure of Category in **BPMN**. However, the two structures can be mapped to each other.*

For example, in a **SDMN** diagram, DataItems can be categorized. The figure below shows how DataItems can be assigned a "Guideline Data" *Category* or a "Referrals" *Category*. In a large **SDMN** diagram, this would allow a modeler to quickly find Data Items of these or other *Categories*.

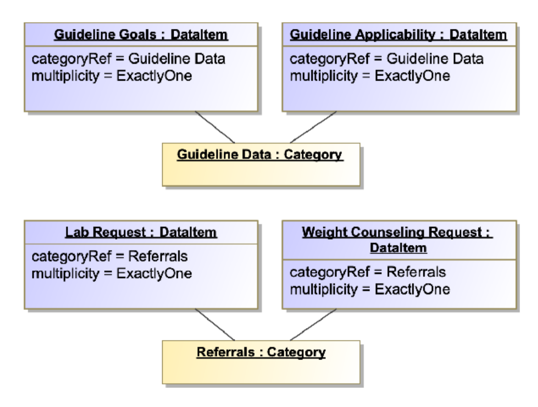

Guideline Goals : DataItem
categoryRef = Guideline Data
multiplicity = ExactlyOne
Guideline Applicability : DataItem
categoryRef = Guideline Data
multiplicity = ExactlyOne
Guideline Data : Category
Lab Request : DataItem
categoryRef = Referrals
multiplicity = ExactlyOne
Weight Counseling Request : DataItem
categoryRef = Referrals
multiplicity = ExactlyOne
Referrals : Category

Figure 6 - An Example of Groups referencing Categories (in an UML Object Diagram)

To support the categorization of model elements, *Categories* can be nested to create a hierarchy of parent and child *Categories*. For example, recommendations can be assigned a *Category* of one of the children of the “Lifestyle Modification” *Category*. As shown in the figure below, the children “Weight Reduction,” “Exercise Program,” and “Diet Modification”. Thus, these Recommendations can be organized under the parent Category and then further organized by the child Categories.

In addition, since a *Category* can reference another *Category*, the Recommendations in the figure below can be identified as being “Patient Responsibilities” through that *Category’s* association with the “Lifestyle Modification” *Category*, which is the parent of the *Category* directly associated with the Recommendation.

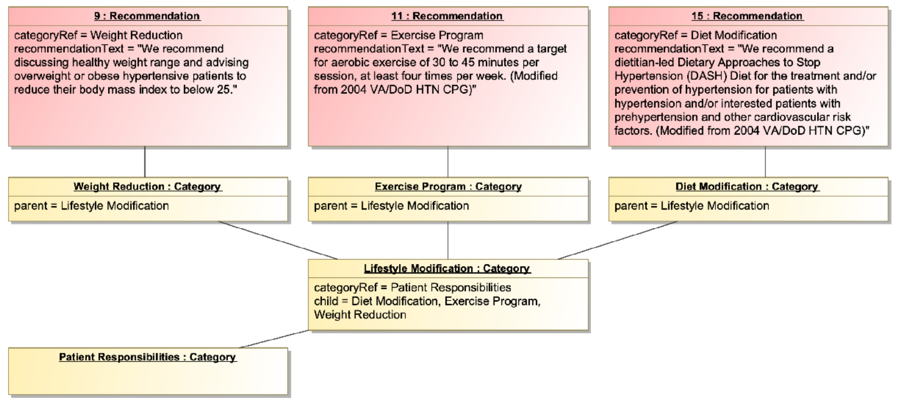

9 : Recommendation
categoryRef = Weight Reduction
recommendationText = "We recommend discussing healthy weight range and advising overweight or obese hypertensive patients to reduce their body mass index to below 25."
11 : Recommendation
categoryRef = Exercise Program
recommendationText = "We recommend a target for aerobic exercise of 30 to 45 minutes per session, at least four times per week. (Modified from 2004 VA/DoD HTN CPG)"
15 : Recommendation
categoryRef = Diet Modification
recommendationText = "We recommend a dietitian-led Dietary Approaches to Stop Hypertension (DASH) Diet for the treatment and/or prevention of hypertension for patients with hypertension and/or interested patients with prehypertension and other cardiovascular risk factors. (Modified from 2004 VA/DoD HTN CPG)"
Weight Reduction : Category
parent = Lifestyle Modification
Exercise Program : Category
parent = Lifestyle Modification
Diet Modification : Category
parent = Lifestyle Modification
Lifestyle Modification : Category
categoryRef = Patient Responsibilities
child = Diet Modification, Exercise Program, Weight Reduction
Patient Responsibilities : Category

Figure 7 - An Example of a Parent and Children Categories (in an UML Object Diagram)

Generalizations
~~Generalizations~~

The *Category* element inherits the attributes and/or associations of:

- *RootElement* (see the section entitled “RootElement” for more information).

Further, the *RootElement* element inherits the attributes and/or associations of:

- *BaseElement* (see the section entitled “BaseElement” for more information).

~~Properties~~

The following table presents the additional attributes and/or associations for *Category*:

~~**Property/Association**~~	~~**Description**~~

~~Table 7.~~Table 6. Category Attributes and/or Associations

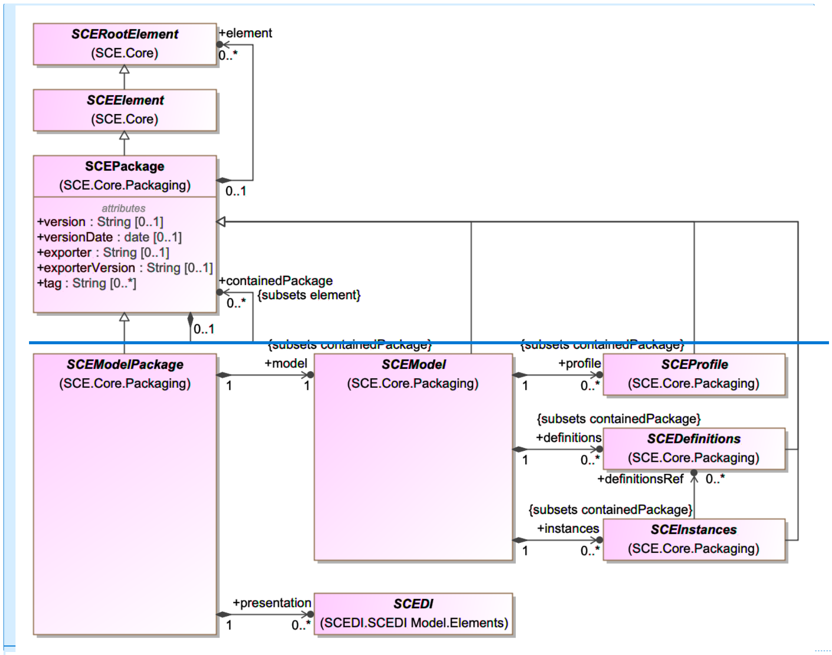
Property/Association	Description
category : Category [0..*]	This association allows the nesting of *Categories*. A *Category* MAY have more than one child *Category*.
parentRef : Category [0..1]	This association allows the nesting of *Categories*. A *Category* MAY be a parent for more than one *Category*.

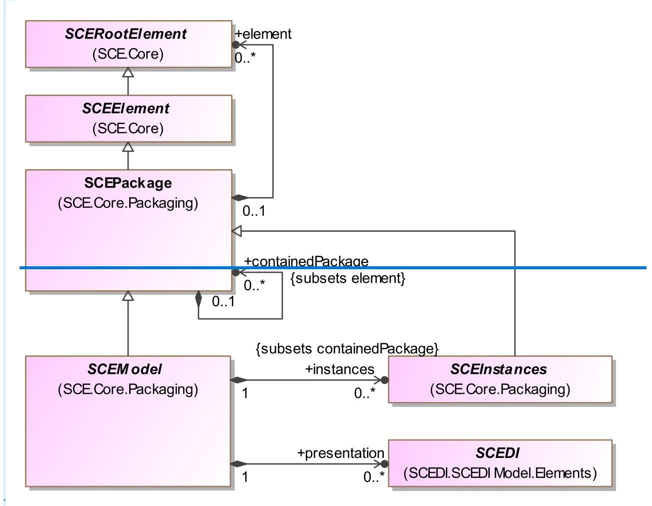
~~8.1.5~~8.1.6Packaging

SCE provides ~~six~~ two elements (*Package* and *Model*) that enable the packaging and distribution of modeling languages dependent on **SCE**. Note that it is not expected that **SCE** “models” will be created and distributed, but the capabilities provided by **SCE** will support the creation and distribution of models created by languages utilizing **SCE**.

The ~~six~~ two sub-sections below will describe the packaging elements provided by **SCE**.

~~The following figure presents the metamodel for **SCE** packaging elements:~~





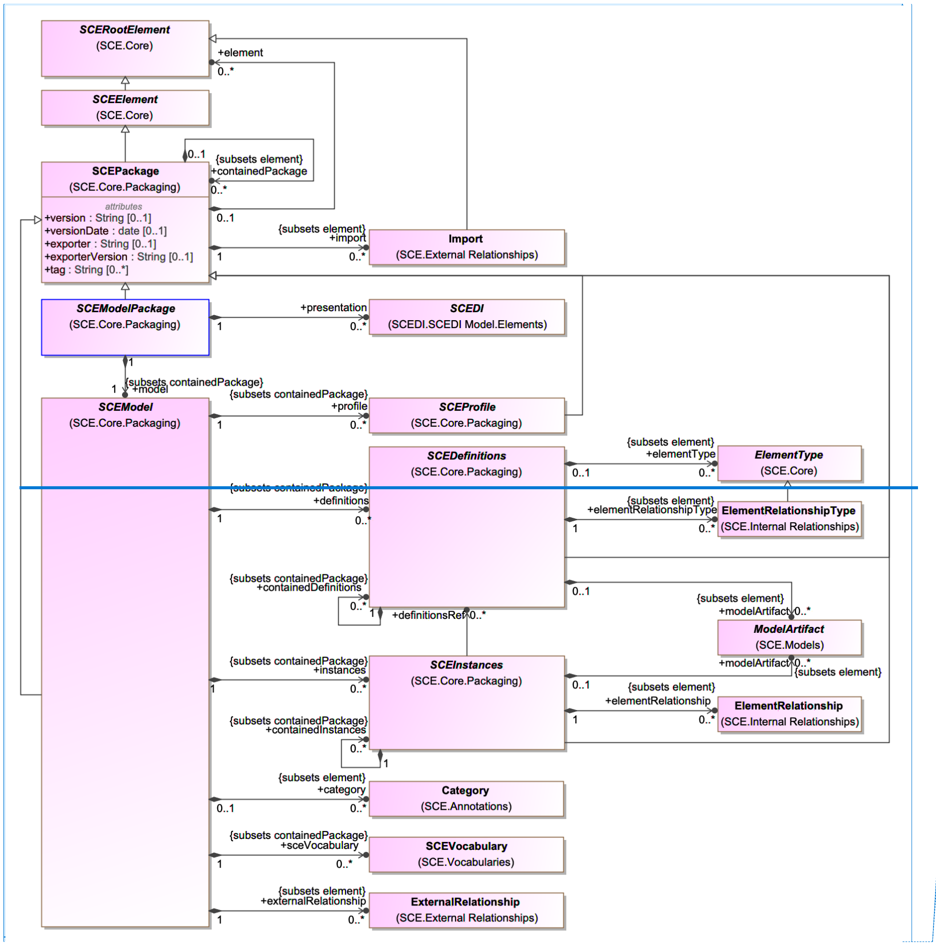
~~Figure 4 - The SCE Packaging Elements Metamodel~~

The following figure presents the attributes and associations for the **SCE** ~~packaging~~ *Package and Model* elements~~, including more details about the elements they contain~~:

Commented [SW59]: This figure was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW60]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW61]: This figure was removed and the text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues. this figure was redundant.

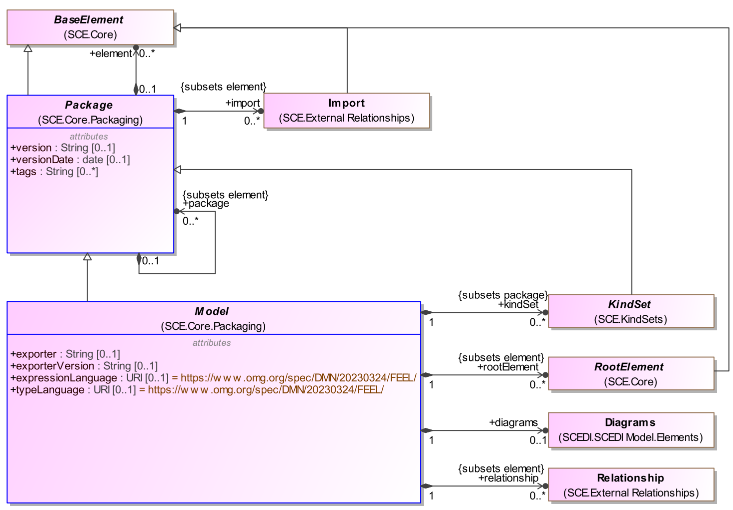


Commented [SW62]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes). The SCEVocabulary class was renamed to SCEKindSet

Commented [SW63]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

Commented [SW64]: This figure was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW65]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.



~~Figure 5 -~~ Figure 8 - The SCE Packaging Elements Metamodel ~~(Details)~~

Commented [SW66]: This figure was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

Commented [SW67]: This figure was also updated for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes. Import no longer a RootElement. RootElement included in Model as a marker class for downstream language elements. Helps with schema ordering. Also ExternalRelationship renamed to Relationship

Commented [SW68]: This figure was also updated for the resolution of SCE-63/SCE-64. The "sce" prefix for the role name for the association to the SCEKindSet was removed. it is now "kindSet".

Commented [SW69]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

Commented [SW70]: This figure was also updated for the resolution of Issue SCE-105/SCE-106. changing tag to tags

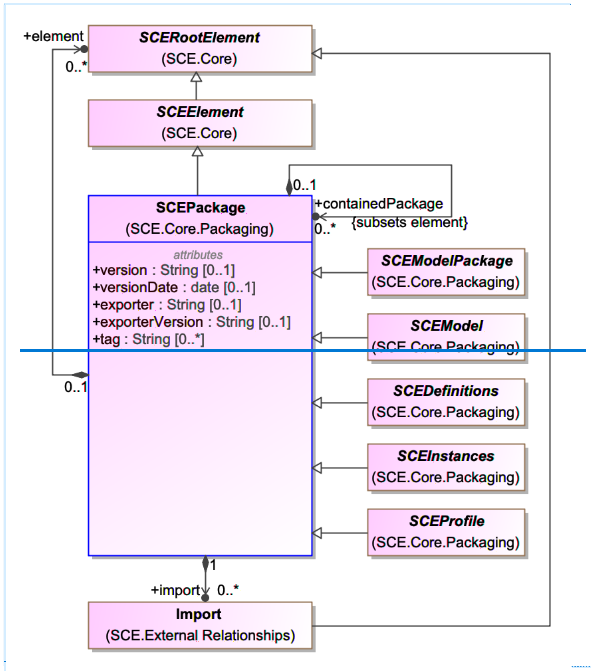
~~8.1.5.1~~8.1.6.1 ~~SCEPackage~~Package

~~SCEPackage~~Package is a basic capability that is used by the other packaging classes in **SCE**. Thus, by itself it is not contained within any element. It's ~~five~~ sub-class, *Model*~~esclasses (listed in the next five sections)~~, will be used to organize the types of content that make up a model or set of models (of a language that utilizes **SCE**). The *~~SCEModel~~Model* *~~SCEModelPackage~~* (see below) is the top-level package used for distribution of the content of a modeling language.

Commented [SW71]: This text was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Note: a `targetNamespce` attribute is not required for the metamodel elements for **SCE**. However, for non-XMI XSDs, a `targetNamespace` attribute of type `anyURI` will be included in the t~~SCEPackage~~Package type for the **SCE** XSD.

~~The following figure presents the metamodel for *SCEPackage*:~~



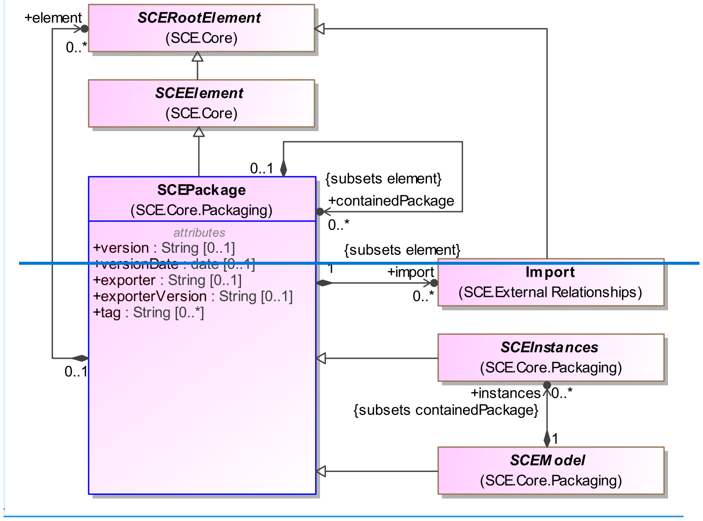


Figure 6 - The SCEPackage Metamodel

Commented [SW72]: This figure was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW73]: This figure was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use. This figure was redundant with the figure above it.

Generalizations

The ~~*SCEPackage*~~*Package* element inherits the attributes and/or associations of:

- ~~*SCEElement*~~*BaseElement* (see the section entitled "~~SCEElement~~BaseElement~~BaseElement~~" for more information).

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

Properties

The following table presents the additional attributes and/or associations for ~~*SCEPackage*~~*Package*:

~~Table 8.~~Table 7. ~~**SCEPackage**~~**Package** **Attributes and/or Associations**

Property/Association	Description
~~**contained**~~**p**~~**P**~~**ackage** : ~~SCEPackage~~Package~~**containedPackage**~~ ~~: SCEPackage~~ [0..*]	This is a list of all the sub-packages ~~*SCEPackage*~~*Package*. This provides the capability for all specializations of ~~*SCEPackage*~~*Package* to include sub-packages. This is a subset of the `element` association of the ~~*SCEPackage*~~*Package* element.
~~**element** : SCERootElement [0..*]~~	~~This is a list of all the *SCERootElementSCERootElements* contained within a *SCEModel*. Many elements will be identified through additional associations that subset this property (see figure above).~~
~~**exporter** : String [0..1]~~	~~This attribute identifies the tool that is exporting the model file that is dependent on **SCE**. If this attribute is specified for a package element and not specified for any of the sub-packages contained within, then the value set for the higher-level package will be assumed for the lower-level packages.~~
~~**exporterVersion** : String [0..1]~~	~~This attribute identifies the version of the tool that is exporting the file that is dependent on **SCE**. If this attribute is specified for a package element and not specified for any of the sub-packages contained within, then the value set for the higher-level package will be assumed for the lower-level packages.~~
import : Import [0..*]	This attribute is used to import externally defined elements and make them available for use by elements within a concrete specialization of an ~~*SCEPackage*~~*Package*.
tags~~**tag**~~ : String [0..*]	The `tag` setting provides another classification mechanism for package. This classification could be used as part of a search for a particular package within a concrete specialization of ~~*SCEModel*~~*Model*~~*SCEModelPackage*~~, for example.
version : String [0..1]	This attribute specifies the version of the model package that is dependent on **SCE**. If this attribute is specified for a package element and not specified for any of the sub-packages contained within, then the value set for the higher-level package will be assumed for the lower-level packages.
versionDate : date [0..1]	The date when the version of the model package that is dependent on **SCE** was established. If this attribute is specified for a package element and not specified for any of the sub-packages contained within, then the value set for the higher-level package will be assumed for the lower-level packages.

Commented [SW74]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SAW75]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW76]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW77]: This text was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW78]: These two rows were removed for the resolution of Issue SCE-85/SCE-86. Move exporter and exporterVersion.

Commented [SAW79]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW80]: This text was updated for the resolution of Issue SCE-105/SCE-106. changing tag to tags

Commented [SW81]: This text was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SAW82]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

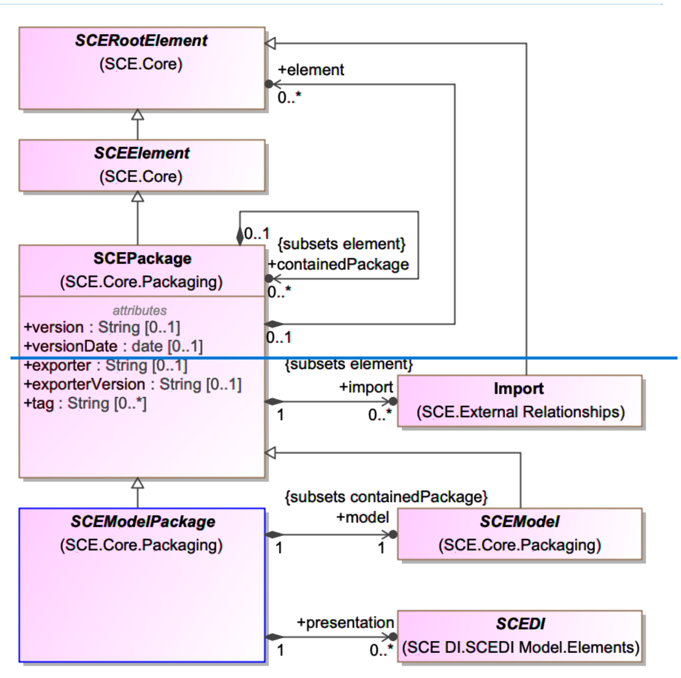
~~8.1.5.2~~8.1.6.2 ~~SCEModel~~Model~~SCEModelPackage~~

This the main **SCE** package, which contains a set of properties and other elements, that are common to and usable by other modeling specifications. The idea of a “package” is that the package will contain all the elements of a model that is based on that specification. When the content of that model is serialized, the elements will be contained within a concrete specialization of *~~SCEModel~~Model.~~SCEModelPackage.~~* Some previous BMI specifications have named this packaging element “Definitions.” In those specifications, they had only one main package that served multiple purposes that **SCE** divided up between its sub-packages. For example, the **BPMN** *Definitions* element is the main package that contains all the Collaborations, Processes, and other elements that make up **BPMN** models, as well as holding the diagram interchange information.

The *~~SCEModel~~Model~~SCEModelPackage~~* element provides the key attributes and associations that most BMI modeling specifications will need as part of their packaging element. **SCE** also provides the capability of a language to define element *instances* and model profiles. To support these additional capabilities, a set of specific sub-packages are defined. Thus, a single “Definitions” top-level package was not sufficient to support the potential languages that will utilize **SCE**.

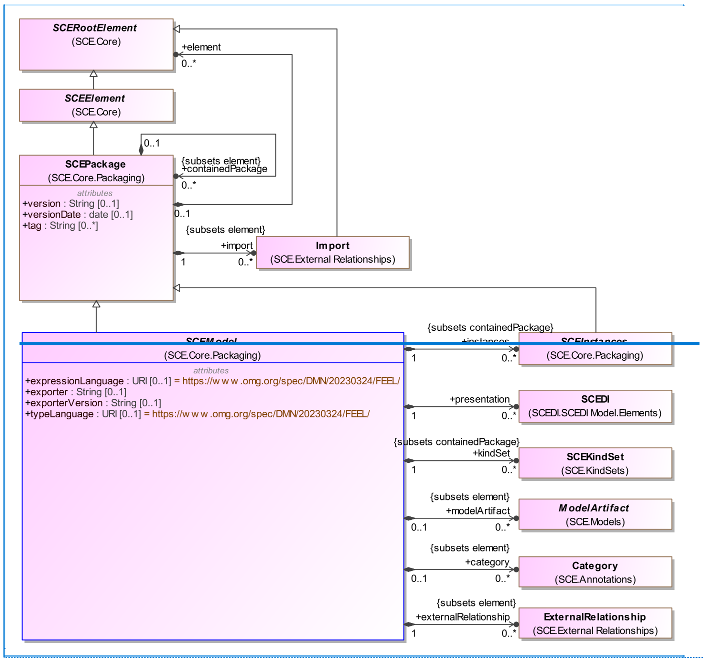
The *~~SCEModel~~Model~~SCEModelPackage~~* element inherits the attributes of *~~SCEPackage~~Package* (see table above). It is an abstract element; thus, **SCE** cannot be implemented by itself to create a modeling package. An implementation of another modeling specification that is dependent on **SCE** is required to produce a concreate modeling package.

~~The following figure presents the metamodel for *SCEModel:SCEModelPackage:*~~



Commented [SW83]: The text in this section updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW84]: This figure was updated as part of the resolution for SCE-2/SCE-52. Added language and type properties to SCEModelPackage



~~Figure 7 - The SCEModelSCEModelPackage Metamodel~~

Generalizations

The ~~*SCEModel*~~*Model*~~*SCEModelPackage*~~ element inherits the attributes and/or associations of:

- ~~*SCEPackage*~~*Package* (see the section entitled "~~SCEPackage~~Package" for more information).

Further, the ~~*SCEPackage*~~*Package* element inherits the attributes and/or associations of:

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement*~~*BaseElement* (see the section entitled "~~SCERootElement~~BaseElement~~BaseElement~~" for more information).

Properties

The following table presents the additional attributes and/or associations for ~~*SCEModel*~~*Model*~~*SCEModelPackage*~~:

Commented [SW85]: This figure was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW86]: This figure was also updated for the resolution of SCE-63/SCE-64. The "sce" prefix for the role name for the association to the SCEKindSet was removed. it is now "kindSet".

Commented [SW87]: This figure was updated for the resolution of Issue SCE-85/SCE-86. Move exporter and exporterVersion.

Commented [SW88]: This figure was also updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

Commented [SW89]: This figure was also updated for the resolution for Issue SCE-67/SCE-78. Setting expressionLanguage and typeLanguage to optional.

Commented [SW90]: This figure was removed for the resolution of Issue SCE-96/SCE-97. More backwards compatibility and simplification of the packaging. Now this figure is redundant with the Packaging metamodel figure above.

Commented [SW91]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

~~Table 9.~~Table 8. ~~SCEModel~~Model~~SCEModelPackageSCEModel~~ Attributes and/or Associations

Property/Association	Description
exporter : String [0..1]	This attribute identifies the tool that is exporting the model file that is dependent on **SCE**. If this attribute is specified for a package element and not specified for any of the sub-packages contained within, then the value set for the higher-level package will be assumed for the lower-level packages.
exporterVersion : String [0..1]	This attribute identifies the version of the tool that is exporting the file that is dependent on **SCE**. If this attribute is specified for a package element and not specified for any of the sub-packages contained within, then the value set for the higher-level package will be assumed for the lower-level packages.
expressionLanguage : URI [0..1] default: https://www.omg.org/spec/DMN/20230324/FEEL/	This attribute identifies the formal expression language used in Expressions within the elements of this ~~*SCEModel*~~*Model*~~*Package*~~. The default is “https://www.omg.org/spec/DMN/20230324/FEEL/”. This value MAY be overridden on each individual formal Expression. The language SHALL be specified in a URI format. If any language that is based on **SCE** requires formal expressions, then that language will have an expression element named *formalExpression* or *literalExpression* or something similar. That element will have to include an optional property named `expressionLanguage` that, if used, will override the default property listed here for ~~*SCEModel*~~*Model*. Thus, every expression defined by that language may use a different expression language (which assumes the target engine can use that language).
~~**externalRelationship**~~**relationship** : ~~ExternalRelationship~~Relationship [0..*]~~**model** : SCEModel [1]~~	~~This is a list of all the *ExternalRelationship*~~*Relationships*~~*SCEModel* sub-package contained within a~~ concrete specialization~~*SCEModelPackage*. This is a subset of the `containedPackage` association of~~ *SCEDefinitions*.~~the *SCEPackage* element.~~
~~**instances** : SCEInstances~~**relationship** : Relationship [0..*] ~~[0..*]~~	This is a list of all the *Relationships* contained within a concrete specialization of *Model*.~~*SCEInstances* sub-packages*SCEModel*. This is a subsetthe containedPackage association of the *SCEPackage* element.~~
~~SCEKindSetterms (*Kinds*) that can be used to define*SCEModel***modelArtifact** : ModelArtifact~~**kindSet** : KindSet [0..1] ~~[0..*]~~	This is a list of terms (*Kinds*) that can be used to define the elements of a concrete specialization of *Model*.~~all*ModelArtifacts* contained within *SCEInstances*. These will usually be contained in an *SCEInstances* that is sub-package to the top-level *SCEInstances*. This is a subset of the `element` association of the *SCEPackage* element~~
rootElement : RootElement [0..1]	This is a list of all the *RootElements* contained within a concrete specialization of *Model*. This is a subset of the `element` association of the *Package* element.
~~**presentation**~~ **diagrams** : ~~SCEDI~~ Diagrams [0..1]	This attribute contains the Diagram Interchange information contained within this ~~*SCEModel*~~*Model*~~*SCEModelPackage*~~.
typeLanguage : URI [0..1] default: https://www.omg.org/spec/DMN/20230324/FEEL/	This attribute identifies the type system used by the elements of this ~~*SCEModel*~~*Model*.~~*Package*~~ The default is “https://www.omg.org/spec/DMN/20230324/FEEL/”. This value can be overridden for each type usage. The language SHALL be specified in a URI format.

Commented [SAW92]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW94]: These two rows were added for the resolution of Issue SCE-85/SCE-86. Move exporter and exporterVersion.

Commented [SW95]: This text was also updated for the resolution for Issue SCE-67/SCE-78. Setting expressionLanguage and typeLanguage to optional.

Commented [SW97]: This table row was added as part of the resolution for SCE-2/SCE-52

Commented [SW96]: This text was added for the resolution for Issue SCE-62/SCE-80. Explain how other expression languages may be used.

Commented [SW98]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. Renaming ExternalRelationship.

Commented [SW99]: This row was removed for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW100]: This row was added for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW101]: This text was also updated for the resolution for Issue SCE-67/SCE-78. Setting expressionLanguage and typeLanguage to optional.

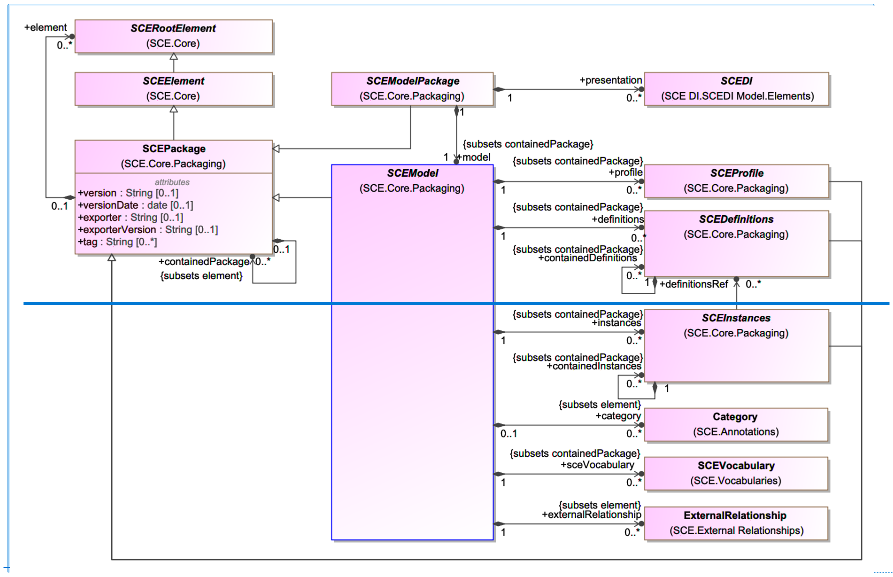
Commented [SW102]: This table row was added as part of the resolution for SCE-2/SCE-52

~~8.1.5.3 SCEModel~~

~~The *SCEModel* is the package that contains most of the **SCE** semantic elements (including model types and instances) and is separate from any diagram information regarding the semantic elements. The *SCEModel* and the *SCEDI* are combined at the top-level *SCEModelPackage*.~~

~~The *SCEModel* element inherits the attributes of *SCEPackage* (see table above). It is an abstract element; thus, **SCE** cannot be implemented by itself to create a modeling package. An implementation of another modeling specification that is dependent on **SCE** is required to produce a concreate modeling package.~~

~~The following figure presents the metamodel for *SCEModel*:~~



~~**Figure 8 - The SCEModel Metamodel**~~

~~The *SCEModel* element inherits the attributes and/or associations of:~~

- ~~*SCEPackage* (see the section entitled "SCEPackage" for more information).~~

~~Further, the *SCEPackage* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

~~The following table presents the additional attributes and/or associations for *SCEModel*:~~

~~**SCEModel**~~	
~~**category** : Category [0..*]~~	~~This is a list of all the *Categories* contained within a concrete specialization of *SCEModel*.~~

Commented [SW104]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes). SCEVocabulary renamed to SCEKindSet

Commented [SW105]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

~~**definitions**: SCEDefinitions [0..*]~~	~~This is a list of all the *SCEDefinitions* sub-packages contained within a *SCEModel*. This is a subset of the `containedPackage` association of the *SCEPackage* element.~~
~~**externalRelationship** : ExternalRelationship [0..*]~~	~~This is a list of all the *ExternalRelationships* contained within a concrete specialization of *SCEDefinitions*.~~
~~**instances** : SCEInstances [0..*]~~	~~This is a list of all the *SCEInstances* sub-packages contained within a *SCEModel*. This is a subset of the `containedPackage` association of the *SCEPackage* element.~~
~~**profile** : SCEProfile [0..*]~~	~~This is a list of all the *SCEProfile* sub-packages contained within a *SCEModel*. This is a subset of the `containedPackage` association of the *SCEPackage* element.~~
~~**sceVocabulary** : SCEVocabulary [0..*]~~	~~This is a list of terms (*SemanticRefernces*) that can be used to define the elements of a concrete specialization of *SCEModel*.~~

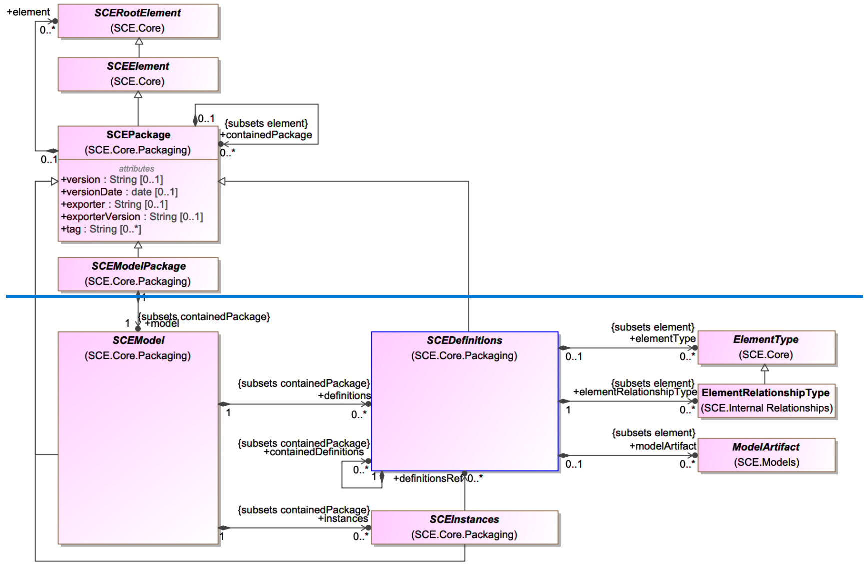
Commented [SW106]: This row was updated for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

~~8.1.5.4 SCEDefinitions~~

~~The *SCEDefinitions* element is the package that, when specialized by a downstream language, will contain the "modeling" elements of that language. In the context of **SDMN** all the modeling elements, such as **Data Items**, would be contained in a specialization of *SCEDefinitions*, such as *SDMNDefinitions* (see below). In the context of **BKPMN** all the modeling elements, such as **ProcessRefs**, would be contained in a specialization of *SCEDefinitions*, such as *BKPMNDefinitions* (see below).~~

~~The *SCEDefinitions* element inherits the attributes of *SCEPackage* (see table above). It is an abstract element; thus, **SCE** cannot be implemented by itself to create a modeling package. An implementation of another modeling specification that is dependent on **SCE** is required to produce a concreate modeling package.~~

~~The following figure presents the metamodel for *SCEDefinitions*:~~



~~**Figure 9 - The SCEDefinitions Metamodel**~~

~~The *SCEDefinitions* element inherits the attributes and/or associations of:~~

- ~~*SCEPackage* (see the section entitled "SCEPackage" for more information).~~

~~Further, the *SCEPackage* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

~~*SCEDefinitions*:~~

~~**Table 10. SCEDefinitions Attributes and/or Associations**~~

~~**Property/Association**~~	~~**Description**~~
~~**containedDefinitions** : SCEDefinitions [0..*]~~	~~This is a list of all the sub-packages *SCEDefinitions*. This provides the capability for all specializations of *SCEDefinitions* to include sub-packages. This is a subset of the `containedPackage` association of the *SCEPackage* element.~~
~~**elementType** : ElementType [0..*]~~	~~This is a list of all the *ElementTypes* contained within a *SCEDefinitions*. This is a subset of the `element` association of the *SCEPackage* element.~~
~~**elementRelationshipType** : ElementRelationshipType [0..*]~~	~~This is a list of all the *ElementRelationshipTypes* contained within a concrete specialization of *SCEDefinitions*. This is a subset of the `element` association of the *SCEPackage* element.~~

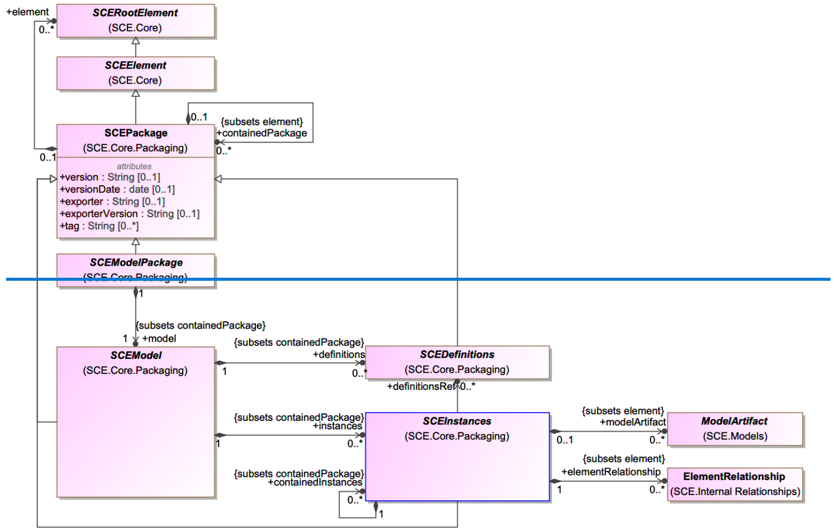
~~**modelArtifact** : ModelArtifact [0..*]~~	~~This is a list of all the *ModelArtifacts* contained within a concrete specialization of *SCEDefinitions*. These will usually be contained in an *SCEDefinitions* that is sub-package to the top-level *SCEDefinitions*. This is a subset of the `element` association of the *SCEPackage* element.~~

~~8.1.5.5 SCEInstances~~

~~The *SCEInstances* element is the package that, when specialized by a downstream language, will contain the specification of the instances of the "modeling" elements of that language. This provides the capability to interchange these instances. Current BPM+ languages, such as **BPMN**, do not formally define the properties or provide for the exchange of their modeling elements (e.g., for a **BPMN** Process instance). **SCE** has been structured to support future languages that formal model the instances. There are at least two specifications in development that will utilize this capability (the Provenance and Pedigree Model and Notation (**PPMN**) and **BKPMN**.~~

~~The *SCEInstances* element inherits the attributes of *SCEPackage* (see table above). It is an abstract element; thus, **SCE** cannot be implemented by itself to create a modeling package. An implementation of another modeling specification that is dependent on **SCE** is required to produce a concreate modeling package.~~

~~The following figure presents the metamodel for *SCEInstances*:~~



~~**Figure 10 - The SCEInstances Metamodel**~~

~~Generalizations~~

~~The *SCEInstances* element inherits the attributes and/or associations of:~~

- ~~*SCEPackage* (see the section entitled "SCEPackage" for more information).~~

~~Further, the *SCEPackage* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

~~Properties~~

~~The following table presents the additional attributes and/or associations for *SCEInstances*:~~

~~Table 11. SCEInstances Attributes and/or Associations~~

~~**Property/Association**~~	~~**Description**~~
~~**containedInstances** : SCEInstances [0..*]~~	~~This is a list of all the sub-packages *SCEInstances*. This provides the capability for all specializations of *SCEInstances* to include sub-packages. This is a subset of the `containedPackage` association of the *SCEPackage* element.~~
~~**definitionsRef** : SCEDefinitions [0..*]~~	~~This is a reference to an *SCEDefinitions* package that contains the *ElementType* elements that provide a basis for the instances contained in the *SCEInstances* package. Note that an *SCEInstances* package is not required to reference a *SCEDefinitions* package.~~
~~**elementRelationship** : ElementRelationship [0..*]~~	~~This is a list of all the *ElementRelationships* contained within a concrete specialization of *SCEDefinitions*. This is a subset of the `element` association of the *SCEPackage* element.~~
~~**modelArtifact** : ModelArtifact [0..*]~~	~~This is a list of all the *ModelArtifacts* contained within a concrete specialization of *SCEInstances*. These will usually be contained in an *SCEInstances* that is sub-package to the top-level *SCEInstances*. This is a subset of the `element` association of the *SCEPackage* element.~~

~~8.1.5.6 SCEProfile~~

~~A kind of *SCEPackage* that comprises **SCE** profiles that can be applied to other **SCE** elements. *SCEProfiles* provide a mechanism to exchange profile libraries.~~

~~The *SCEProfile* element inherits the attributes of *SCEPackage* (see table above). It is an abstract element; thus, **SCE** cannot be implemented by itself to create a modeling package. An implementation of another modeling specification that is dependent on **SCE** is required to produce a concreate modeling package.~~

~~Generalizations~~

~~The *SCEProfile* element inherits the attributes and/or associations of:~~

- ~~*SCEPackage* (see the section entitled "SCEPackage" for more information).~~

~~Further, the *SCEPackage* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

~~Properties~~

~~The *SCEProfile* element does not have any additional attributes and/or associations.~~

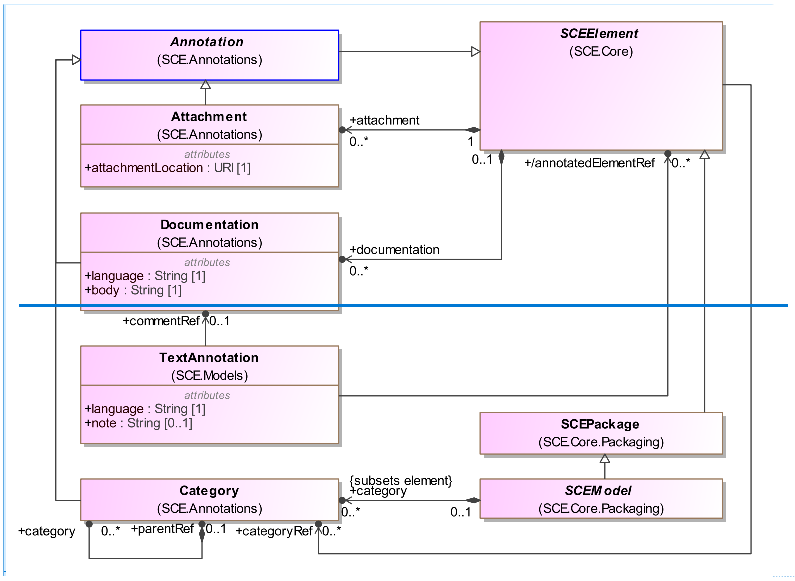
Commented [SW108]: This table row was deleted for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW110]: This section was deleted for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

8.2 Annotations

Annotations allow information, provided by a modeler of a modeling language that is dependent on **SCE**, to be attached to a *~~SCEElement~~BaseElement*-based element order document or categorize that element. This attached information is generally for the benefit of readers or users of the model that contains the annotated element. There are currently ~~three~~ two concrete types of *Annotations*: *Attachments~~, Categories,~~* and *Documentation*.

The following figure shows the metamodel for *Annotations*~~.~~:

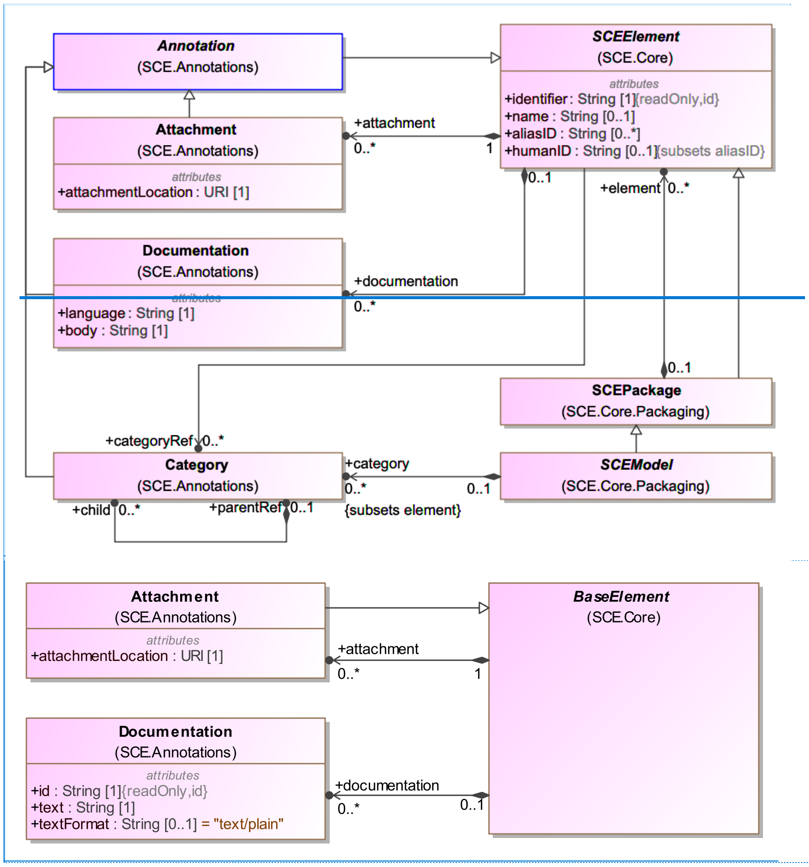


Commented [SW111]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW112]: This text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues.

Commented [SW113]: Text updated as a resolution for SCE-27/SCE-30 (fix typos)

Commented [SW114]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.



~~Figure 11 -~~ Figure 9 - Annotations

~~8.2.1 Annotation~~

~~The *Annotation* element is an abstract element that is used to organize a set of elements that are used to annotate any concrete specialization of *SCEElement*. The containment of *Annotations* depends on the specific type of *Annotation* (see the next three sections).~~

~~Generalizations~~

~~The *Annotation* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

~~Properties~~

Commented [SW115]: Figure updated as resolution for SCE-28/SCE-29 (incorrect metamodel figure).

Commented [SW116]: This text was also updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. Category was moved to a RootElement. TextAnnotations was removed. Documentation attributes were updated.

Commented [SW117]: This figure was updated for the resolution of Issue SCE-35/SCE-81. including a change to the Category association role names.

Commented [SW118]: This figure was also updated resolution of Issue SCE-107/SCE-108. Removing Annotation Class.

Commented [SW119]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

~~The *Annotation* element does not have any additional attributes and/or associations.~~

Commented [SW120]: This section was removed resolution of Issue SCE-107/SCE-108. Removing Annotation Class.

~~8.2.2~~8.2.1 Attachment

The *Attachment* element provides a place for model developers to provide attached documents to a model element.

The *Attachment* element is contained within a concrete specialization of *~~SCEElement~~BaseElement*. Thus, any concrete element within a model that is dependent on **SCE** MAY have one or more *Attachments*.

Commented [SAW121]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Generalizations

The *Attachment* element inherits the attributes and/or associations of:

- ~~*Annotation* (see the section entitled "Annotation" for more information).~~

~~Further, the *Annotation* element inherits the attributes and/or associations of:~~

Commented [SW122]: This text was removed resolution of Issue SCE-107/SCE-108. Removing Annotation Class.

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- *~~SCERootElement~~BaseElement* (see the section entitled "~~SCERootElement~~BaseElement~~SCERootElement~~" for more information).

Commented [SW123]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for *Attachment*:

~~Table 12.~~Table 9. Attachment Attributes and/or Associations

Property/Association	Description
attachmentLocation : URI [1]	This attribute identifies the URI location of the attachment.

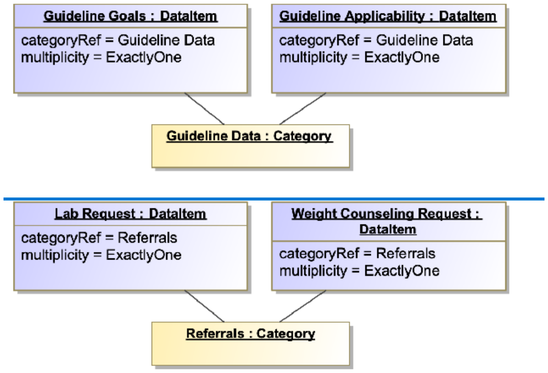
~~8.2.3 Category~~

~~A *Category*, which have user-defined semantics, can be used for documentation or metadata organizational purposes. For example, recommendations (in the healthcare domain) can be assigned a category of "Lifestyle Modification" with further breakdowns into "Weight Reduction," "Exercise Program," and "Diet Modification" sub-categories.~~

~~The *Category* element inherits the attributes of *SCEElement* (see table above) and is contained within a *SCEModel* (see figure above). It is referenced by any *SCEElement*. Thus, any concrete element within a model file, dependent on **SCE**, MAY have zero or more *Categories*. Further, *Categories* may be nested such that one *Category* may contain other *Categories*.~~

> ~~*Note: The structure of Category in **SCE** is different than the structure of Category in **BPMN**. However, the two structures can be mapped to each other.*~~

~~For example, in a **SDMN** diagram, Data Items can be categorized. The figure below shows how Data Items can be assigned a "Guideline Data" *Category* or a "Referrals" *Category*. In a large **SDMN** diagram, this would allow a modeler to quickly find Data Items of these or other *Categories*.~~

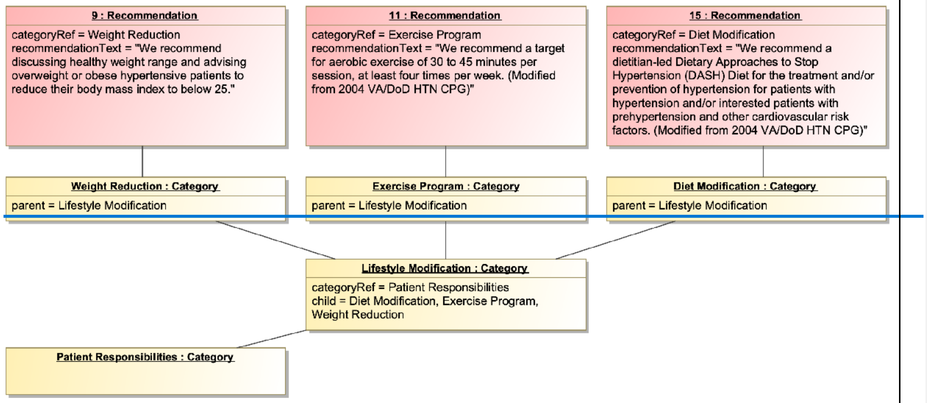

Guideline Goals : DataItem
categoryRef = Guideline Data
multiplicity = ExactlyOne
Guideline Applicability : DataItem
categoryRef = Guideline Data
multiplicity = ExactlyOne
Guideline Data : Category
Lab Request : DataItem
categoryRef = Referrals
multiplicity = ExactlyOne
Weight Counseling Request : DataItem
categoryRef = Referrals
multiplicity = ExactlyOne
Referrals : Category

~~Figure 12 - An Example of a Groups referencing Categories (in an UML Object Diagram)~~

Commented [SW124]: This text has been updated as a resolution for Issue SCE-9/SCE-65. Fix typos in Category section.

~~To support the categorization of model elements, *Categories* can be nested to create a hierarchy of parent and child *Categories*. For example, in a **BKPMN** BPM+ Knowledge Package, recommendations can be assigned a *Category* of one of the children of the "Lifestyle Modification" *Category*. As shown in the figure below, the children "Weight Reduction," "Exercise Program," and "Diet Modification". Thus, these Recommendations can be organized under the parent Category and then further organized by the child Categories.~~

~~In addition, since a *Category* can reference another *Category*, the Recommendations in the figure below can be identified as being "Patient Resonsibilities" through that *Category's* association with the "Lifestyle Modification" *Category*, which is the parent of the *Category* directly associated with the Recommendation.~~


9 : Recommendation
categoryRef = Weight Reduction
recommendationText = "We recommend discussing healthy weight range and advising overweight or obese hypertensive patients to reduce their body mass index to below 25."
11 : Recommendation
categoryRef = Exercise Program
recommendationText = "We recommend a target for aerobic exercise of 30 to 45 minutes per session, at least four times per week. (Modified from 2004 VA/DoD HTN CPG)"
15 : Recommendation
categoryRef = Diet Modification
recommendationText = "We recommend a dietitian-led Dietary Approaches to Stop Hypertension (DASH) Diet for the treatment and/or prevention of hypertension for patients with hypertension and/or interested patients with prehypertension and other cardiovascular risk factors. (Modified from 2004 VA/DoD HTN CPG)"
Weight Reduction : Category
parent = Lifestyle Modification
Exercise Program : Category
parent = Lifestyle Modification
Diet Modification : Category
parent = Lifestyle Modification
Lifestyle Modification : Category
categoryRef = Patient Responsibilities
child = Diet Modification, Exercise Program, Weight Reduction
Patient Responsibilities : Category

~~Figure 13 - An Example of a Parent and Children Categories (in an UML Object Diagram)~~

Generalizations

~~The *Category* element inherits the attributes and/or associations of:~~

- ~~*Annotation* (see the section entitled "Annotation" for more information).~~

~~Further, the *Annotation* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

~~Properties~~

~~The following table presents the additional attributes and/or associations for *Category*:~~

~~Table 13. Category Attributes and/or Associations~~

~~**Property/Association**~~	~~**Description**~~
~~**child** : Category [0..*]~~	~~This association allows the nesting of *Categories*. A *Category* MAY have more than one child *Category*.~~
~~**parentRef** : Category [0..1]~~	~~This association allows the nesting of *Categories*. A *Category* MAY be a parent for more than one *Category*.~~

Commented [SW125]: This text was updated for the resolution of Issue SCE-35/SCE-81. including a change to the Category association role names.

~~8.2.4~~8.2.2Documentation

The *Documentation* element provides a place for model developers to provide descriptive information about a~~n~~ model element.

The *Documentation* element is contained within a concrete specialization of *~~SCEElement~~BaseElement*. Thus, any concrete element within a model that is dependent on **SCE** MAY have one or more *Documentations*.

Generalizations

The *Documentation* element does not inherit the attributes and/or associations of another element.

~~The *Documentation* element inherits the attributes and/or associations of:~~

- ~~*Annotation* (see the section entitled "Annotation" for more information).~~

~~Further, the *Annotation* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElementSCERootElement" for more information).~~

Commented [SAW127]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for *Documentation*:

~~Table 14.~~Table 10. Documentation Attributes and/or Associations

Property/Association	**Description**
id : String [0..1]	This optional attribute is used to uniquely identify a *Documentation*. The `id` is REQUIRED if this element is referenced or intended to be referenced by something else. If the element is not currently referenced and is never intended to be referenced, the `id` MAY be omitted.
~~**body**~~ **text** : String [1]	This attribute is used to capture the text descriptions of any concrete element within a model that is dependent on **SCE**.

textFormat : String [0..1] = "text/plain"	This attribute identifies the format of the text. It MUST follow the mime-type format. The default is "text/plain."
~~**language** : String [1]~~	~~The named language can be a natural language, in which case the body is an informal representation, or an artifical language, in which case the body is expected to be a formal, machine-parsable representation.~~

Commented [SW128]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. Documentation attributes updated.

8.3 External Relationships

*Note: the text and metamodel defined in this section are based on the External Relationships definitions found in the **BPMN** specification.*

BPM+ models do not exist in isolation and generally participate in larger, more complex business and system development efforts. The intention of the following specification element is to enable BPM+ models to be integrated in these development efforts via the specification of a non-intrusive identity/relationship model between BPM+ models and elements expressed in any other addressable domain model.

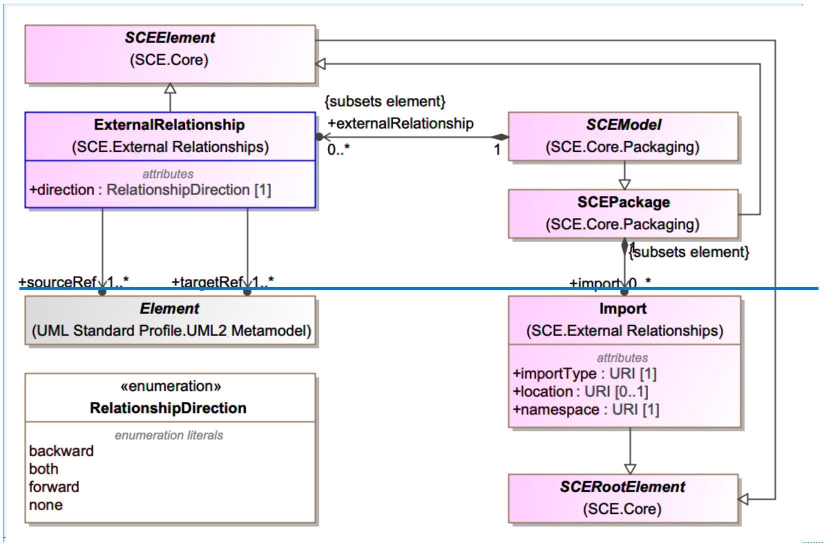
The 'identity/relationship' model it is reduced to the creation of families of typed relationships that enable BPM+ and non-BPM+ Artifacts to be related in non-intrusive manner. By simply defining 'relationship types' that can be associated with elements in the BPM+ Artifacts and arbitrary elements in a given addressable domain model, it enables the extension and integration of BPM+ models into larger system/development efforts.

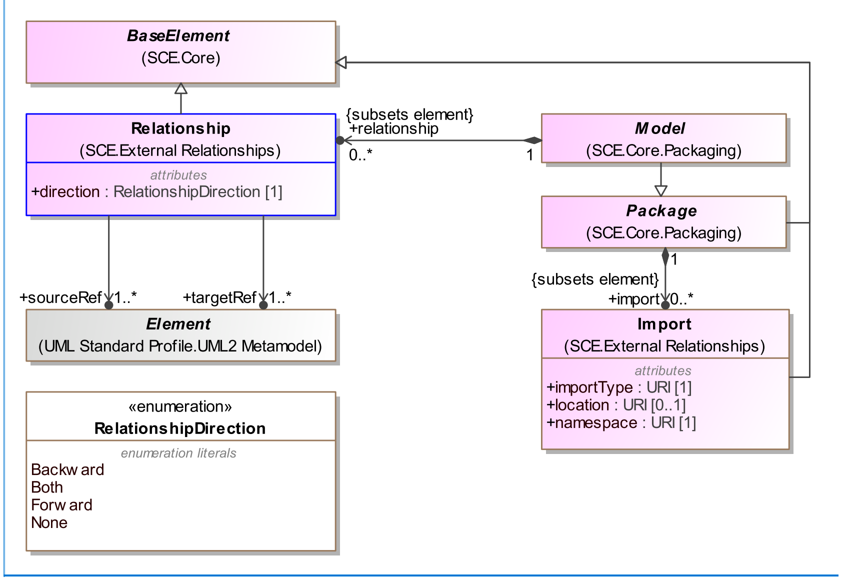
It is that these extensions will enable, for example, the linkage of 'derivation' or 'definition' relationships between UML artifacts and BPM+ Artifacts in novel ways. So, a UML use case could be related to a BPM+ element in a specification dependent on **SCE** without affecting the nature of the Artifacts themselves but enabling different integration models that traverse specialized relationships.

Simply, the model enables the external specification of augmentation relationships between BPM+ Artifacts and arbitrary relationship classification models, these external models, via traversing relationships declared in the external definition allow for linkages between BPM+ elements and other structured or non-structured metadata definitions.

The following figure shows the *~~External~~Relationship* metamodel diagram.

Commented [SW129]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. Renaming ExternalRelationship.





~~Figure 14 -~~ Figure 10 - The External Relationships Metamodel

8.3.1 ~~ExternalRelationship~~Relationship

The ~~*ExternalRelationship*~~*Relationship* element is where an external relationship can be defined. It allows a relationship to be defined between and internal model element and an external model element. It is contained in a~~n~~

Commented [SW130]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW131]: This figure was also updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. Renaming of Element to BaseElement

~~SCEModel~~Model.

Generalizations

The *~~ExternalRelationship~~Relationship* element inherits the attributes and/or associations of:

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- *~~SCERootElement~~BaseElement* (see the section entitled "BaseElement~~BaseElementSCERootElement"~~"SCERootElement" for more information).

Properties

The following table presents the additional attributes and/or associations for *~~ExternalRelationship~~Relationship*:

~~Table 15.~~Table 11. ~~ExternalRelationship~~Relationship Attributes and/or Associations

Property/Association	Description
direction : RelationshipDirection [1]	This attribute specifies the direction of the external relationship. See the *RelationshipDirection* enumeration, below, for more details.
sourceRef : Element [1..*]	This association defines artifacts that are augmented by the external relationship.
targetRef : Element [1..*]	This association defines artifacts used to extend the semantics of the source element(s).

Commented [SAW132]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

8.3.2 RelationshipDirection

This enumeration list specifies the direction of the relationship.

The following table lists and defines the *RelationshipDirection* literals.

~~Table 16.~~Table 12. RelationshipDirection Literals

Literal	Description
B~~b~~ackward	This literal specifies that the *~~ExternalRelationship~~Relationship* is in the direction from the `target` to the `source`.
B~~b~~oth	This literal specifies that the *~~ExternalRelationship~~Relationship* is in the direction from the `target` to the `source` and from the `source` to the `target`.
F~~f~~orward	This literal specifies that the *~~ExternalRelationship~~Relationship* is in the direction from the `source` to the `target`.
N~~n~~one	This literal specifies that the *~~ExternalRelationship~~Relationship* is in the direction from the `target` to the `source`.

Commented [SAW133]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

8.3.3 Import

The *Import* class is used by an implementation of a modeling specification (i.e., a model), dependent on **SCE**, when referencing an external element that is contained in a different model. The referenced model can be of the same or different type of modeling specification. It is contained within a concrete specialization of *~~SCEPackage~~Package*.

Generalizations

The *Import* does not inherit ~~element inherits~~ the attributes and/or associations of another element~~:~~.~~:~~

- ~~*SCERootElement* (see the section entitled "SCERootElementSCERootElement" for more information).~~

Commented [SAW134]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for *Import*:

~~Table 17.~~Table 13. Import Attributes and/or Associations

Property/Association	Description
importType : URI [1]	Identifies the type of document being imported by providing an absolute URI that identifies the encoding language used in the document, e.g. when importing XML Schema 1.0 documents the value of the importType attribute MUST be set to http://www.w3.org/2001/XMLSchema. Other types of documents MAY be supported, e.g. BPMN, CMMN, DMN or any SCE-based language.~~Identifies the type of document being imported by providing an absolute URI that identifies the encoding language used in the document. The value of the importType attribute MUST be set to http://www.w3.org/2001/XMLSchema when importing XML Schema 1.0 documents, to http://www.w3.org/TR/wsdl20/ when importing WSDL 2.0 documents, and http://www.omg.org/spec/BPMN/20100524/MODEL when importing BPMN 2.0 documents. Other types of documents MAY be supported. Importing Xml Schema 1.0, WSDL 2.0 and BPMN 2.0, CBMN 1.0, CMMN 1.1, DMN 1.5types5types35, and SDMN 1.0 types MUST be supported. Further, any BPM+ specification that is developed that utilizes the SCE infrastucture MUST be supported.~~ ~~Identifies the type of document being imported by providing an absolute URI that identifies the encoding language used in the document. The value of the importType attribute MUST be set to http://www.w3.org/2001/XMLSchema when importing XML Schema 1.0 documents, to http://www.w3.org/TR/wsdl20/ when importing WSDL 2.0 documents, and http://www.omg.org/spec/BPMN/20100524/MODEL when importing BPMN 2.0 documents. Other types of documents MAY be supported. Importing Xml Schema 1.0, WSDL 2.0 and BPMN 2.0, CBMN 1.0, CMMN 1.1, DMN 1.3, and SDMN 1.0 types MUST be supported.~~
location : ~~URI~~ string [0..1]	Identifies the location of the imported element within the document identified by the `importType`.
namespace : URI [1]	Identifies the namespace of the imported element.

Commented [SW135]: This text was updated for the resolution of Issue SCE-101-/SCE-102. Update importType property description.

Commented [SW136]: This text was updated for the resolution of Issue SCE-36/SCE-76. Update the Import description.

Commented [SW137]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

8.4 Internal Relationships

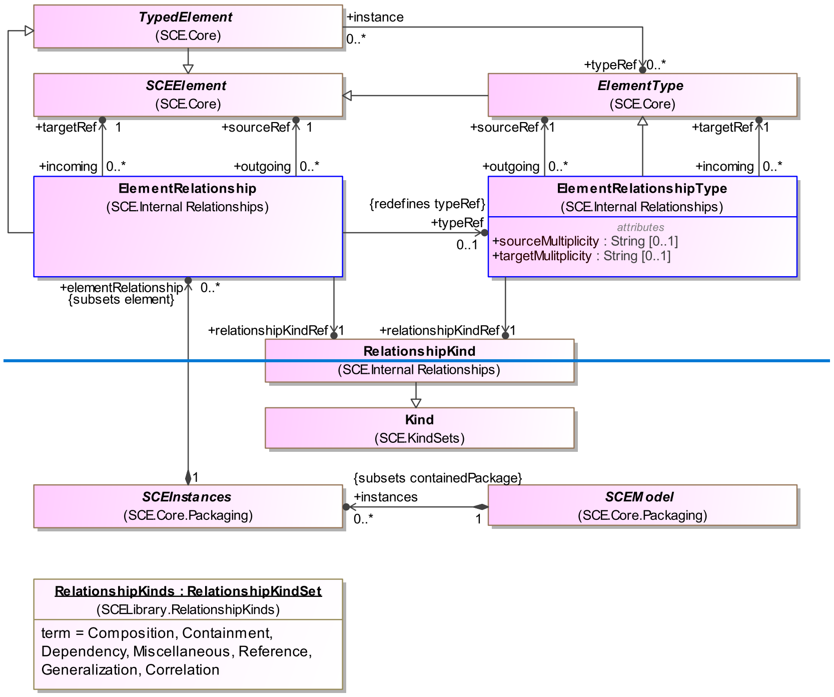
The intention of the following specification element is to enable BPM+ models to develop relationships between modeling elements within a specific language. Most of these types of relationships will be specific to the context of a modeling language that is dependent on **SCE**.

The following figure presents the metamodel for *ElementRelationship* and *ElementRelationshipType* (including the predefined instance of ~~*SCEKindSet*~~*KindSet*~~*SDMNVocabulary*~~~~*SDMNVocabulary*~~ for *RelationshipKind*):

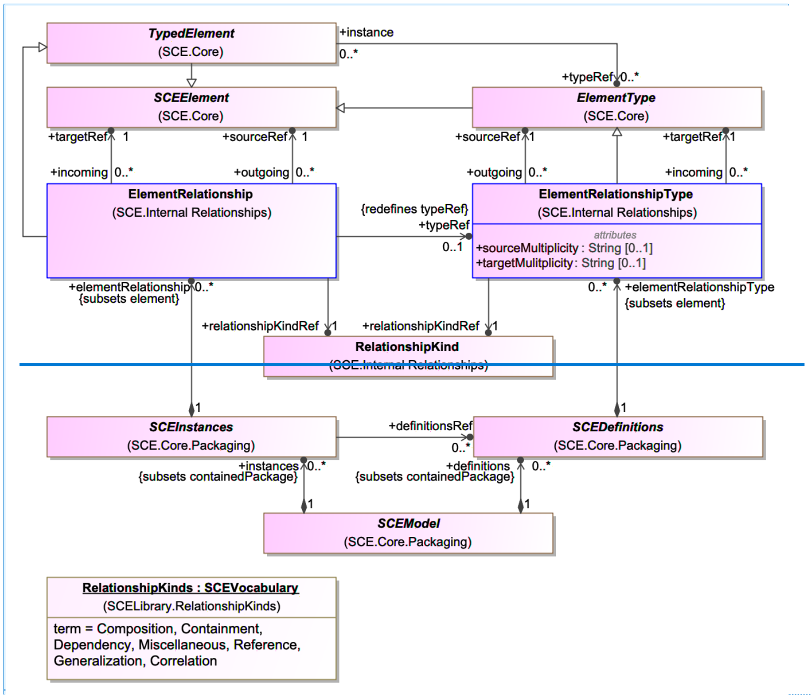
Commented [SW138]: The following text was updated for the resolution of Issue SCE-90/SCE-95. Remove SCE Prefix

Commented [SW139]: this text was changed for Issue SCE-7/SCE-8 (Semantic Reference changes)

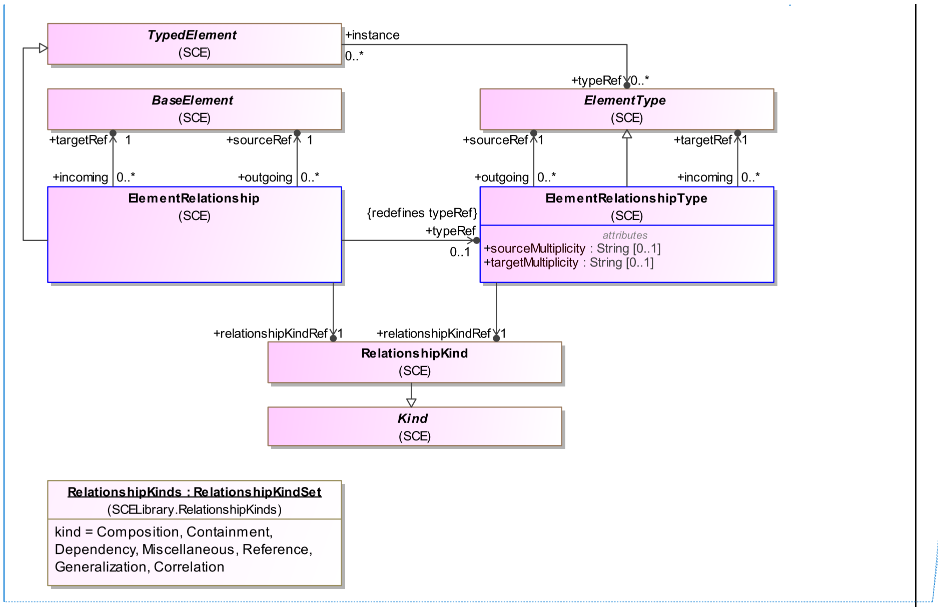
~~Figure 15 -~~



Commented [SW140]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.



Commented [SW141]: This figure was updated for Issue SCE-7/SCE-8 (Semantic Reference changes). Kind identified as superclass for RelationshipKind. RelationshipKinds instance reset to RelationshipKindSet (a new class)



Commented [SW142]: This figure was also updated for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes.

Commented [SW143]: This figure was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW144]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

~~Figure 16 -~~ Figure 11 - The Internal Relationships Metamodel

8.4.1 ElementRelationship

A kind of relationships between two *~~SCEElement~~BaseElements.~~SCEElements.~~* The *RelationshipKind* element (see the section "RelationshipKind", below for more information) identify specific types of relationships.

Commented [SW145]: This text was added for the resolution of Issue SCE-7/SCE-8 (Semantic Reference and vocabulary changes)

Generalizations

The *ElementRelationship* element inherits the attributes and/or associations of:

- *TypedElement* (see the section entitled "TypedElement~~TypedElement~~" for more information).

Further, the *TypedElement* element inherits the attributes and/or associations of:

- *~~SCEElement~~RootElement* (see the section entitled "RootElement~~SCEElement~~"SCEElement" for more information).

Further, the *~~SCEElement~~RootElement* element inherits the attributes and/or associations of:

- *~~SCERootElement~~BaseElement* (see the section entitled "BaseElement~~BaseElement~~SCERootElement"SCERootElement" for more information).

Commented [SW146]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for *ElementRelationship*:

~~Table 18.~~Table 14. ElementRelationship Attributes and/or Associations

Property/Association	Description
sourceRef : ~~SCEElement~~BaseElement [1]	The source ~~*SCEElement*~~*BaseElement* of the relationship. If there is an *ElementRelationshipType* identified through the `typeRef` association, then the source must be a *TypedElement*.
targetRef : ~~SCEElement~~BaseElement [1]	The target concrete specialization of ~~*SCEElement*~~*BaseElement* of the relationship. If there is an *ElementRelationshipType* identified through the `typeRef` association, then the target must be a *TypedElement*.
relationshipKindRef : RelationshipKind [1]	A description of the type of the relationship. See *RelationshipKind*, below, for more details.
typeRef : ElementRelationshipType [0..1]	The class(es) that provide(s) a specification of the *ElementRelationship*. This usually is applied to the concrete *ElementRelationshipType* that serves as an instance in a runtime model. This redefines the `typeRef` association of *TypedElement*.

Commented [SAW147]: This text was updated for the resolution of Issue SCE-121/SCE-122. Replace Element with BaseElement in table.

8.4.2 ElementRelationshipType

A kind of *ElementRelationship* that specifies two *ElementTypes* (rather than ~~*SCEElement*~~*BaseElements*).~~*SCEElements*).~~ The *RelationshipKind* element identify specific types of relationships.

Generalizations

The *ElementRelationshipType* element inherits the attributes and/or associations of:

- *ElementType* (see the section entitled "Element~~Element~~Type" for more information).

Further, the *ElementType* element inherits the attributes and/or associations of:

- ~~*SCEElement*~~*RootElement* (see the section entitled "RootElement~~SCEElement~~~~"SCEElement~~" for more information).

Further, the ~~*SCEElement*~~*RootElement* element inherits the attributes and/or associations of:

- ~~*SCERootElement*~~*BaseElement* (see the section entitled "BaseElement~~BaseElementSCERootElement~~~~"SCERootElement~~" for more information).

Commented [SW148]: This text was updated from the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. merging RootElement and Element.

Properties

The following table presents the additional attributes and/or associations for *ElementRelationshipType*:

~~Table 19.~~Table 15. ElementRelationshipType Attributes and/or Associations

Property/Association	Description
sourceMultiplicity : String [0..1]	This attribute defines the minimum number of source ~~*SCEElement*~~*BaseElements*~~*SCEElements*~~ that may be the source for the *ElementRelationship* that identifies this *ElementRelationshipType* through its `typeRef` association.
sourceRef : ElementType [1]	The source *ElementType* of the relationship.

targetMultiplicity : String [0..1]	This attribute defines the minimum number of target *~~SCEElement~~BaseElements~~SCEElements~~* that may be the source for the *ElementRelationship* that identifies this *ElementRelationshipType* through its `typeRef` association.
targetRef : ElementType [1..*]	The one or more target *ElementTypes* of the relationship.
relationshipKindRef: RelationshipKind [1]	A description of the type of the relationship. See *RelationshipKind*, below, for more details.

Commented [SAW149]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

~~**Table 20.**~~

~~**Instance**~~	~~**Description**~~
~~**Composition**~~	~~`Composition` indicates that the `source` element is composed of, in part, the `target` element. Other elements could be included in this composition.~~
~~**Containment**~~	~~`Containment` indicates that the `source` element is a container for the `target` element.~~
~~**Correlation**~~	~~`Correlation` indicates that the `source` element is correlated with the `target` element. This is often used when a mapping is required between the structures of two data elements.~~
~~**Dependency**~~	~~`Dependency` indicates that `target` element is dependent in some way on the `source` element.~~
~~**Miscellaneous**~~	~~`Miscellaneous` indicates that `source` element has some relationship with the `target` element that is of a kind that is not expressed through the other *RelationshipKind* instances.~~
~~**Reference**~~	~~`Reference` indicates that `source` element references the `target` element.~~
~~**Generalization**~~	~~`Generalization` indicates that the `source` element is a generalization of the `target` element (which is based on and extends the `source`).~~

8.5 BPM+ Modeling

The main purpose of BPM+ modeling specifications is to provide the languages for business analysts to create specific *models* (that the language defines). For example, **BPMN** defines Process models, Collaboration models, etc; and **CMMN** defines Case models. **SCE** does not define any specific semantic element since that is the responsibility of the specific BPM+ specification. However, **SCE** provides a basic foundation for models for the modeling languages that utilize **SCE**. BPM+ Modeling languages will include, and perhaps extend, the **SCE** *ModelArtifacts* (see next section) within the *models* defined by those languages.

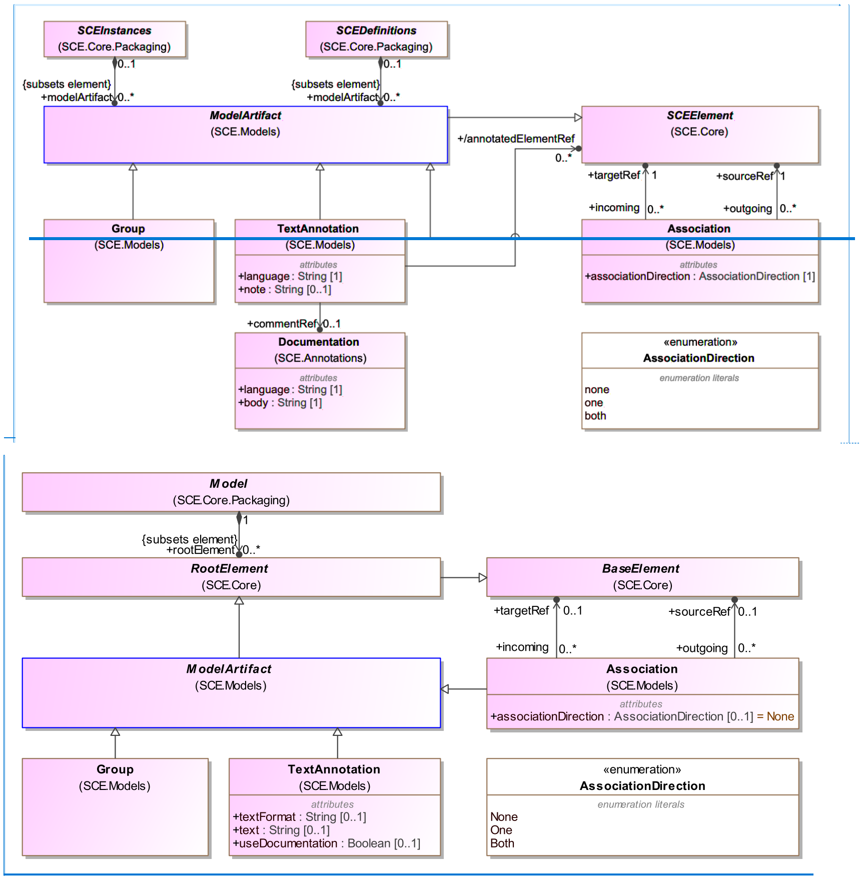
8.5.1 ModelArtifact

A *ModelArtifact* is an object that provides supporting information about a model. However, it does not have any behavioral semantics. The *ModelArtifact* element is an abstract element that inherits the attributes of *~~SCEElement~~BaseElement*. *ModelArtifacts* are contained within a model type that is defined by a modeling language that extends **SCE**. ~~This will usually be a concrete specialization of a sub-package for *SCEDefinitions* or a sub-package for *SCEInstances*.~~

Commented [SW150]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

At this point, **SCE** provides three standard Artifacts: **Associations**, **Groups**, and **Text Annotations**. Additional Artifacts MAY be added to the **SCE** specification in later versions. A modeler or modeling tool MAY extend a model and add new types of *ModelArtifacts*. Any new *ModelArtifacts* MUST follow the connector connection rules defined in the modeling specification that is dependent on **SCE**. **Associations** can be used to link *ModelArtifacts* to model elements and other *ModelArtifacts*.

The following figure shows the *ModelArtifact* metamodel diagram.



~~Figure 17 -~~ Figure 12 - The ModelArtifact Metamodel

Commented [SW151]: This figure was updated for the resolution of Issue SCE-40/SCE-41. Restructured the packaging for backwards capability, including deleting SCEProfile, SCE Definitions, and SCEModel. SCEModelPackage was renamed to SCEModel.

Commented [SW152]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW153]: This figure was also updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW154]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

Generalizations

The *ModelArtifact* element inherits the attributes and/or associations of:

- *RootElement* (see the section entitled "RootElement" for more information).

Further, the *RootElement* element inherits the attributes and/or associations of:

- ~~*SCEElement* (see the section entitled “SCEElement” for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement*~~*BaseElement* (see the section entitled “BaseElement~~BaseElementSCERootElement“SCERootElement~~” for more information).

Commented [SW155]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The *ModelArtifact* element does not have any additional attributes and/or associations.

8.5.2 Association

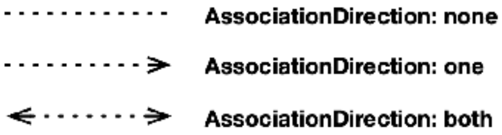
An **Association** is used to associate *ModelArtifacts* (often **Text Annotations**) to other diagram elements. If a *ModelArtifact* extension, such as an image, is added to the model, then that new *ModelArtifact* can be connected by an **Association**. A modeler can set the direction ~~direct~~ of the association such that the connector line will have an arrowhead on either one end or both (see figure below). The presence of one or two arrowheads does not have any specific semantic meaning but may provide a visual queue about the nature of the association.

Commented [SW156]: This text was updated as a resolution for Issue SCE-16/SCE-17

As a *ModelArtifact*, an **Association** is contained within a model type that is defined by a modeling language that extends **SCE**.

Notation

- An **Association** is a line that MUST be drawn with a dotted single line (see figure below) and MAY have a line arrowhead, if needed.
  - The use of text, color, size, and lines for an **Association** MUST follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram” on Page 4.
- If there is a reason to put directionality on the **Association**, then:
  - A line arrowhead MAY be added to the **Association** line (see below).
  - The directionality of the **Association** can be in one direction or in both directions.


AssociationDirection: none
AssociationDirection: one
AssociationDirection: both

~~Figure 18 -~~ Figure 13 - An Association

An **Association** is used to connect user-defined text (a **Text Annotation**) with a diagram element (see figure below).

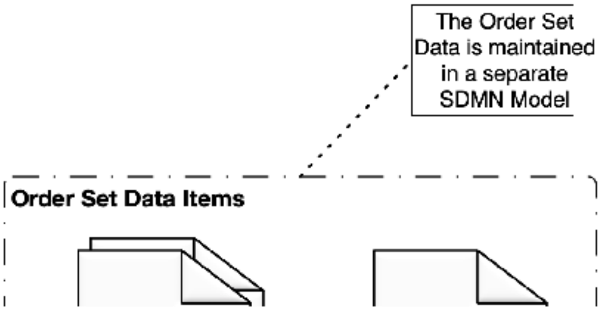


Figure 14 - An Association Used with a Text Annotation

~~**Figure 19 -**~~

Connection Rules

The following statements define connection rules for an **Association** (when used by a modeling language dependent on **SCE**):

- The source of an **Association** MAY be any diagram element (either a *ModelArtifact* or the semantic diagram elements of the modeling language using the **Association**).
- The target of an **Association** MAY be any diagram element (either a *ModelArtifact* or the semantic diagram elements of the modeling language using the **Association**).

Generalizations

The **Association** element inherits the attributes and/or associations of:

- *ModelArtifact* (see the section entitled “ModelArtifact~~ModelArtifact~~” for more information).

Further, the *ModelArtifact* element inherits the attributes and/or associations of:

- *RootElement* (see the section entitled “RootElement” for more information).

Further, the *RootElement* element inherits the attributes and/or associations of:

- ~~*SCEElement* (see the section entitled “SCEElement” for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement*~~*BaseElement* (see the section entitled “BaseElement~~BaseElement~~~~SCERootElement~~“~~SCERootElement~~” for more information).

Commented [SW157]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for **Association**:

~~Table 21.~~Table 16. Association Attributes and/or Associations

Property/Association	Description
associationDirection : AssociationDirection [0..1] = “None”	`AssociationDirection` is an attribute that defines whether or not the **Association** shows any directionality with an arrowhead. The default is “N~~n~~one” (no arrowhead). A value of “O~~o~~ne” means that the arrowhead SHALL be at the target object. A value of “B~~b~~oth” means that there SHALL be an arrowhead at both ends of the **Association** line.
sourceRef : ~~SCEElement~~BaseElement [0..~~[~~1]	The *~~SCEElement~~BaseElement* that the **Association** is connecting from.
targetRef : ~~SCEElement~~BaseElement [0..~~[~~1]	The *~~SCEElement~~BaseElement* that the **Association** is connecting to.

Commented [SW158]: This text was updated for the resolution of SCE-96/SCE-97. More work for backwards compatibility.

8.5.3 AssociationDirection

AssociationDirection is an enumerated list that defines the options regarding whether or not an **Association** shows any directionality with an arrowhead. The default is “none” (no arrowhead). A value of “one” means that the arrowhead SHALL be at the target object. A value of “both” means that there SHALL be an arrowhead at both ends of the **Association**.

The following table lists and defines the *AssociationDirection* literals.

~~Table 22.~~Table 17. AssociationDirection Literals

Literal	Description
B~~b~~oth	A value of “B~~b~~oth” means that there SHALL be an arrowhead at both ends of the **Association**.
N~~n~~one	The default is “N~~n~~one” (no arrowhead).
O~~o~~ne	A value of “O~~o~~ne” means that the arrowhead SHALL be at the *targetRef* Object.

Commented [SW159]: The following text was updated for the resolution of Issue SCE-96/SCE-97. more work for backwards compatibility. capitalizing literals

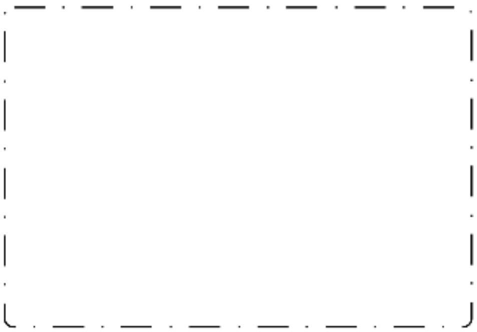
8.5.4 Group

The **Group** object is a *ModelArtifact* that provides a mechanism to informally group elements of a model. **Groups** are often used to highlight certain sections of a model without adding additional constraints or semantics. The highlighted (grouped) section of the model can be separated for reporting and analysis purposes.

As a *ModelArtifact*, a **Group** is contained within a model type that is defined by a modeling language that extends **SCE**.

Notation

- A **Group** is a rounded corner rectangle that MUST be drawn with a solid dashed and dotted line (as seen in the figure below).
    - The use of text, color, size, and lines for a **Group** MUST follow the rules defined in the section entitled “Use of Text, Color, Size, and Lines in a Diagram”, above.



~~Figure 20 -~~ Figure 15 - A Group

Generalizations

The **Group** element inherits the attributes and/or associations of:

- *ModelArtifact* (see the section entitled "ModelArtifact" for more information).

Further, the *ModelArtifact* element inherits the attributes and/or associations of:

- *RootElement* (see the section entitled "RootElement" for more information).

Further, the *RootElement* element inherits the attributes and/or associations of:

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement*~~*BaseElement* (see the section entitled "BaseElement~~BaseElementSCERootElement"~~SCERootElement" for more information).

Commented [SW160]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The **Group** element does not have any additional attributes and/or associations.

8.5.5 TextAnnotation

TextAnnotations are a mechanism for a modeler to provide additional information for the reader of a model.

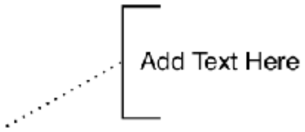
As a *ModelArtifact*, a **TextAnnotation** is contained within a model type that is defined by a modeling language that extends **SCE**.

Notation

- A **Text Annotation** is an open rectangle that MUST be drawn with a solid single line (as seen in Figure 8.16).
    - The use of text, color, size, and lines for a **Text Annotation** MUST follow the rules defined in the section entitled "Use of Text, Color, Size, and Lines in a Diagram", above.
- The **Text Annotation** object can be connected to a specific object on the diagram with an **Association**.
    - The `associationDirection` of the **Association** MUST be "none."

Note that the ***Association*** *is not required for a* ***Text Annotation****. That is, the* ***Text Annotation*** *can be "floating" on a diagram.*

- Text associated with the **Text Annotation** MUST be placed within the bounds of the open rectangle.



~~Figure 21 -~~ Figure 16 - A Text Annotation

Generalizations

The **TextAnnotation** element inherits the attributes and/or associations of:

- *~~DiagramArtifact~~ ModelArtifact* (see the section entitled "~~DiagramArtifact~~ModelArtifact~~DiagramArtifact~~" for more information).

Commented [SW161]: This text was removed for the resolution of Issue SCE-69/SCE-106. Editorial Changes.

Further, the *DiagramArtifact* element inherits the attributes and/or associations of:

- *RootElement* (see the section entitled "RootElement" for more information).

Further, the *RootElement* element inherits the attributes and/or associations of:

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- *~~SCERootElement~~BaseElement* (see the section entitled "BaseElement~~BaseElement~~SCERootElement~~"SCERootElement~~" for more information).

Commented [SW162]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for **TextAnnotation**:

~~Table 23.~~Table 18. TextAnnotation Attributes and/or Associations

Property/Association	Description
~~note~~ text : String [0..1]	~~Note~~ text is one of two attributes that provides text that the modeler wishes to communicate to the reader of the diagram. The text within a ~~note~~ text is contained in and specific to the diagram where the **TextAnnotation** is placed. This attribute is optional, but if it used, then the ~~commentRef~~ useDocumentation attribute SHALL NOT be used.
textFormat : String [0..1] = "text/plain"	This attribute identifies the format of the text. It MUST follow the mime-type format. The default is "text/plain." This attribute is optional, but if useDocumentation used, then this attribute SHALL NOT be used.

useDocumentation : boolean [0..1] = false	`useDocumentation` is one of two attributes that provides text that the modeler wishes to communicate to the reader of the model. This flag will allow a **TextAnnotation** to display the *Documentation* of the model element that the **TextAnnotation** is associated with, i.e., is connected to by an **Association**. This attribute is optional, but if it used, then the `text` and `textformat` attribute SHALL NOT be used. This MUST not be true if there is no **Association**.

Commented [SW163]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. TextAnnotation were modified for better connection to Documentation and to make the attributes consistent with Documentation's attributes.

8.5.6 Diagram Artifact Connection Rules

A modeling specification that is dependent on **SCE** will define connection rules that determine how *DiagramArtifacts* are used within the diagrams defined in that specification. In general, *DiagramArtifacts* are kept separate from the semantic elements and behaviors of the diagrams. **Associations** can be used to create non-semantic connections between the diagrams semantic elements and *DiagramArtifacts*.

8.6 ~~Vocabularies~~KindSets

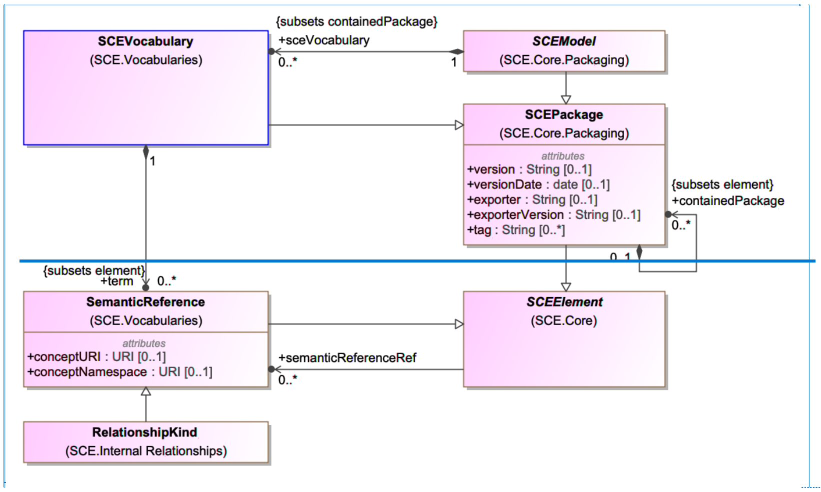
Commented [SW164]: Section title changed for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

Commented [SAW165]: This text was updated from the resolution of Issue SCE-90/SCE-95. removing SCE prefix

~~*SCEKindSet*~~*KindSets*~~Vocabularies (lists of terms) can be added to a model package of a modeling language dependent on **SCE**. *SCEVocabularies*~~ are *Kinds* (terms) ~~sets of terms defined by an external ontology~~that that make up an extendable enumerated list of values for a text-based property of an element property. This capability is included in **SCE** for enumerated lists that should not be fixed by a particular version of **SCE** or a BPM+ language dependent on **SCE**.~~.~~

The terms can link to formal definitions for the model elements that are created by the modeling language. The ~~*SemanticReference*~~ *Kind* element is used to name the term provide a link to the definitions. ~~*SCEKindSet*~~*KindSets* ~~*SCEVocabularies*~~ are contained within a~~n~~ ~~*SCEModel*~~*Model* package. Specific *KindSets* can be established for languages utilizing SCE by creating sub-classes for the ~~*SCEKindSet*~~*KindSet* and *Kind* classes.

The following figure presents the metamodel for ~~*SCEKindSet*~~*KindSet*:~~*SCEVocabularySCEVocabulary*:~~



Commented [SW166]: These paragraphs were updated for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

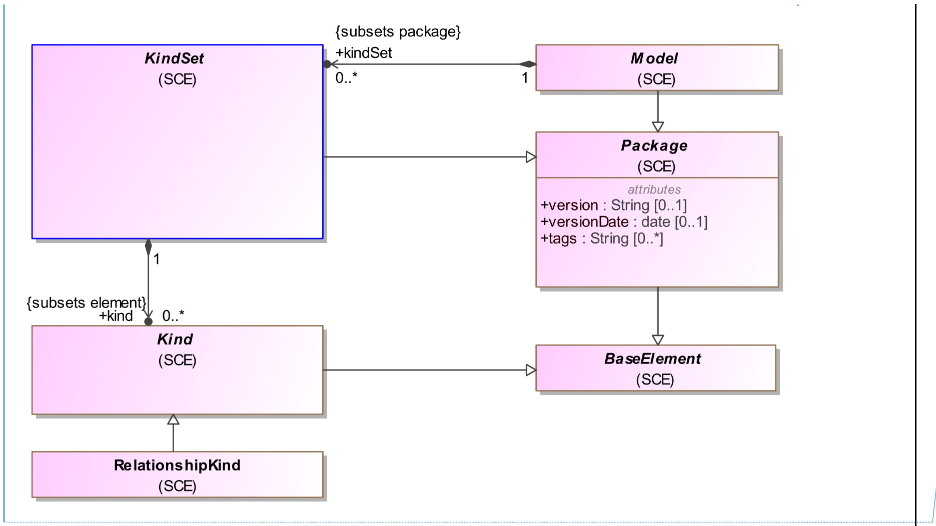
Commented [SW167]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes). SCEVocabulary was renamed to SCEKindSet. SemanticReference was renamed Kind. Kind was reset to be a subclass of SCERootElement not SCEElement.

Commented [SW168]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

Commented [SW169]: This figure was updated for the resolution of SCE-63/SCE-64. The "sce" prefix for the role name for the association to the SCEKindSet was removed. it is now "kindSet".

Commented [SW170]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW171]: This figure was also updated for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes.



~~Figure 22 -~~ Figure 17 - The ~~SCEKindSet~~KindSet ~~SCEVocabulary~~ Metamodel

Commented [SW172]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

Commented [SW173]: This figure was also updated for the resolution of Issue SCE-105/SCE-106. changing tag to tags

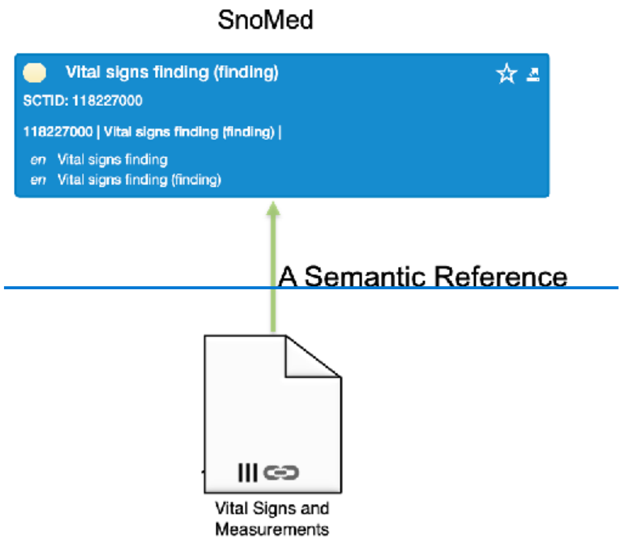
~~8.6.1 SemanticReference~~

~~Most BPM+ models (dependent on **SCE**) are not intended to define full-scale ontologies or domain models, such as data models. However, the activities, decisions, data items, etc. of BPM+ are representative of elements defined by ontologies or data models. The specific context of the BPM+ elements may result in different terminology or sub-sets of data representation elements within the normative domain models. To reduce any confusion due to terminology or data representation, the BPM+ models dependent on **SCE** have the capability of linking model elements to the appropriate external sources of truth for their domain. The *SemanticReference* is that mechanism in **SCE**. It is contained within a *SCEVocabulary* and can be referenced by any *SCEElement*. This means that any model element from a specification dependent on *SCEElement*, directly or indirectly, may include one or more *SemanticReferences*.~~

~~The following figure shows the concept of linking a **SDMN** Data Item to external reference that provides an agreed upon definition of the concept represented by the Data Item. In this example, a "Vital Signs and Measurements" Data Item is linked to an item named "Vital signs finding (finding)" in SnoMed, which is a health care domain site that provides accepted definitions of health care concepts. Note that **SDMN** *does not* show this relationship graphically.~~

Commented [SW174]: This section was removed for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes). SemanticReference was renamed to Kind - and new section for Kind has been added below.

Commented [SW175]: This section was removed for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)


SnoMed
Vital signs finding (finding)
SCTID: 118227000
118227000 | Vital signs finding (finding) |
en Vital signs finding
en Vital signs finding (finding)
A Semantic Reference
Vital Signs and Measurements

~~Figure 23 - An Example of a Semantic Reference within a SDMN Model~~

~~Generalizations~~

~~The *SemanticReference* element inherits the attributes and/or associations of:~~

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement* (see the section entitled "SCERootElement" for more information).~~

~~Properties~~

~~The following table presents the additional attributes and/or associations for *SemanticReference*:~~

~~Table 24. SemanticReference Attributes and/or Associations~~

~~**Property/Association**~~	~~**Description**~~
~~**conceptNamespace** : URI [0..1]~~	~~This attribute documents the version of the target of the *SemanticReference* when the *SemanticReference* was included in the model.~~ ~~If this information is not provided, then it is likely that the conceptURI will navigate to the current version of the target of the *SemanticReference*, which could have changed since the *SemanticReference* was established in the model.~~
~~**conceptURI** : URI [0..1]~~	~~This attribute defines the URI location of the target of the *SemanticReference*.~~

~~8.6.2~~8.6.1~~SCEKindSet~~~~SCEVocabulary~~KindSet

Commented [SW176]: this header was changed for Issue SCE-7/SCE-8 (Semantic Reference and vocabulary changes)

A~~n~~ ~~*SCEKindSetSCEVocabulary*~~ *KindSet* is a list of terms, through the ~~*SemanticReference*~~ *Kind* element, that can be used to relate to model elements to the external definition or meaning. The terms themselves do not represent the definitions or meanings but provide links to an external source. Multiple ~~*SCEKindSet*~~*KindSets*~~*SCEVocabularies*~~ can be defined. They are contained in a~~n~~ ~~*SCEModel*~~*Model*.

Further, ~~*SCEKindSet*~~*KindSets*~~*SCEVocabularies*~~ can be used for creating a user-defined list of enumerated values for use within a modeling language (as opposed to a fixed enumeration list). It is up to the modeling language using **SCE** to organize the ~~*SCEKindSet*~~*KindSets* ~~*SCEVocabularies*~~ into the appropriate enumerated lists. Since the ~~*SemanticReference*~~ *Kind* element has a name and the links to external definitions are optional, the list (the "enumeration" ~~*SCEKindSet*~~*KindSet*~~*SCEVocabulary*~~) can be created before the specific external definitions are established.

SCE has one pre-defined ~~*SCEKindSet*~~*KindSet* ~~*SCEVocabulary*~~ for the enumerated terms for the *RelationshipKind* element (see the section entitled "RelationshipKind~~RelationshipKind~~" for more information).

Commented [SW177]: the text in these three paragraphs was changed for Issue SCE-7/SCE-8 (Semantic Reference and vocabulary changes)

Generalizations

The ~~*SCEKindSet*~~*KindSet*~~*SCEVocabulary*~~ element inherits the attributes and/or associations of:

- ~~*SCEPackage*~~*Package* (see the section entitled "~~SCEPackage~~Package" for more information).

Further, the ~~*SCEPackage*~~*Package* element inherits the attributes and/or associations of:

Commented [SW178]: This text was updated for the resolution of Issue SCE-90/SCE-95. removing SCE prefix.

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement*~~*BaseElement* (see the section entitled "BaseElement~~BaseElement~~~~SCERootElement~~~~"SCERootElement~~" for more information).

Commented [SW179]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

Commented [SW180]: This text was updated and the table removed for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW181]: this section and table was changed for Issue SCE-7/SCE-8 (Semantic Reference and vocabulary changes)

The following table presents the additional attributes and/or associations for *KindSet*:

Table 19. KindSet Attributes and/or Associations

Property/Association	Description
term : Kind [0..*]	The list of terms is a set of *Kinds* which can be linked to an external ontology through the *Kinds'* `conceptReference` property.

8.6.2 Kind

Commented [SW182]: This section was added for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

~~SCERootElement~~A *Kind* is one of a set of *Kinds* (terms) for a *KindSet* that make up an extendable enumerated list of values for a text-based property of an element property. A *KindSet* is a list of terms, through the *Kind* element, that can be used to relate to model elements to the external definition or meaning. The terms themselves do not represent the definitions or meanings but can provide links to an external source.

Commented [SW183]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Generalizations

The *Kind* element inherits the attributes and/or associations of:

- *BaseElement* (see the section entitled "BaseElement~~BaseElement~~" for more information).

Properties

The *Kind* element does not have any additional attributes and/or associations.

8.6.3 RelationshipKindSet

An *RelationshipKindSet* is a list of terms, through the *RelationshipKind* element, that is used to define a set of relationship terms (the relationship between two elements in a model). The terms themselves do not represent the definitions or meanings but provide links to an external source. They are contained in a *Model.*

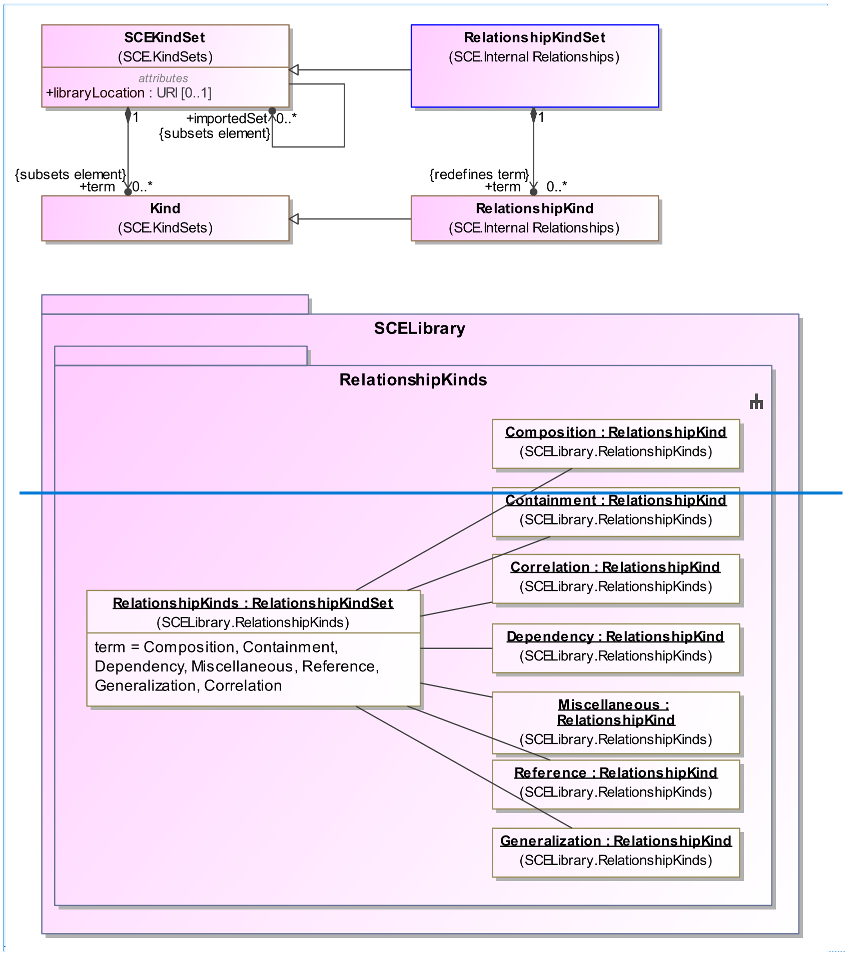
Further, a *RelationshipKindSet* can be used for extending the list of enumerated values in the *RelationshipKinds* Library (see below) for use within a modeling language (as opposed to a fixed enumeration list). It is up to the modeling language using **SCE** to organize the *RelationshipKindSet* into the appropriate enumerated lists. Since the *RelationshipKind* element has a name and the links to external definitions are optional, the list (the “enumeration” *RelationshipKindSet*) can be created before the specific external definitions are established.

SCE has one pre-defined *RelationshipKinds* Library for the enumerated terms for the *RelationshipKind* element (see the section entitled “RelationshipKinds” for more information).

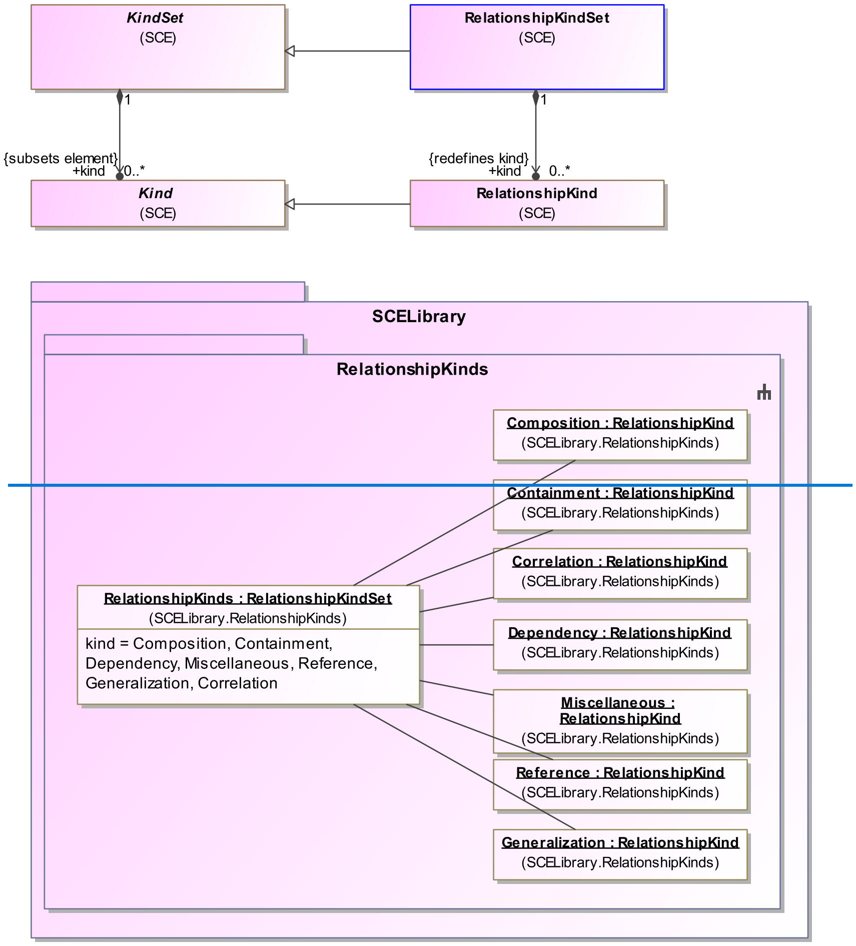
The following figure shows the *RelationshipKindSet* metamodel diagram (which includes the standard set of instances provided by the **SCE** Library).

Commented [SW184]: This section was added for the resolution of Issue SCE-96/SCE-97. Additional backwards compatibility changes. RootElement and Element merged and renamed to BaseElement. A new RootElement added.

Commented [SW185]: This section was added for Issue SCE-7/SCE-8 (Semantic Reference and vocabulary changes)


SCEKindSet
(SCE.KindSets)
attributes
+libraryLocation : URI [0..1]
RelationshipKindSet
(SCE.Internal Relationships)
1
+importedSet
0..*
{subsets element}
1
{subsets element}
+term
0..*
{redefines term}
+term
0..*
Kind
(SCE.KindSets)
RelationshipKind
(SCE.Internal Relationships)
SCELibrary
RelationshipKinds
Composition : RelationshipKind
(SCELibrary.RelationshipKinds)
Containment : RelationshipKind
(SCELibrary.RelationshipKinds)
Correlation : RelationshipKind
(SCELibrary.RelationshipKinds)
RelationshipKinds : RelationshipKindSet
(SCELibrary.RelationshipKinds)
term = Composition, Containment, Dependency, Miscellaneous, Reference, Generalization, Correlation
Dependency : RelationshipKind
(SCELibrary.RelationshipKinds)
Miscellaneous : RelationshipKind
(SCELibrary.RelationshipKinds)
Reference : RelationshipKind
(SCELibrary.RelationshipKinds)
Generalization : RelationshipKind
(SCELibrary.RelationshipKinds)

Commented [SW186]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.


KindSet
(SCE)
RelationshipKindSet
(SCE)
1
1
{subsets element}
+kind
0..*
{redefines kind}
+kind
0..*
Kind
(SCE)
RelationshipKind
(SCE)
SCELibrary
RelationshipKinds
Composition : RelationshipKind
(SCELibrary.RelationshipKinds)
Containment : RelationshipKind
(SCELibrary.RelationshipKinds)
Correlation : RelationshipKind
(SCELibrary.RelationshipKinds)
RelationshipKinds : RelationshipKindSet
(SCELibrary.RelationshipKinds)
kind = Composition, Containment, Dependency, Miscellaneous, Reference, Generalization, Correlation
Dependency : RelationshipKind
(SCELibrary.RelationshipKinds)
Miscellaneous : RelationshipKind
(SCELibrary.RelationshipKinds)
Reference : RelationshipKind
(SCELibrary.RelationshipKinds)
Generalization : RelationshipKind
(SCELibrary.RelationshipKinds)

Commented [SW187]: Figure replaced for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes). SCEVocabulary renamed to SCEKindSet. SemanticReference renamed to Kind. RelationshipKindSet added and used for the RelationshipKinds library instance.

Commented [SW188R187]: This figure was also moved from the RelationshipKind section for the same resolution.

Commented [SW189]: This figure was also updated resolution of Issue SCE-103/SCE-104. Setting concrete elements to abstract.

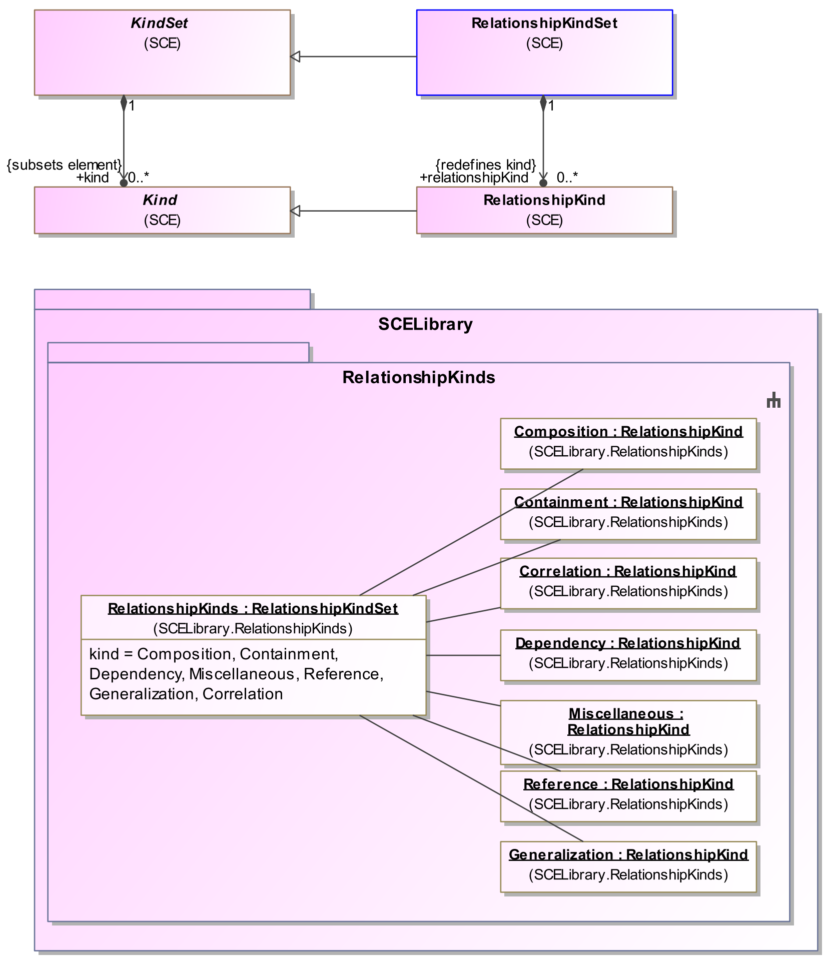

KindSet
(SCE)
RelationshipKindSet
(SCE)
1
1
{subsets element}
+kind
0..*
{redefines kind}
+relationshipKind
0..*
Kind
(SCE)
RelationshipKind
(SCE)
SCELibrary
RelationshipKinds
Composition : RelationshipKind
(SCELibrary.RelationshipKinds)
Containment : RelationshipKind
(SCELibrary.RelationshipKinds)
Correlation : RelationshipKind
(SCELibrary.RelationshipKinds)
RelationshipKinds : RelationshipKindSet
(SCELibrary.RelationshipKinds)
kind = Composition, Containment, Dependency, Miscellaneous, Reference, Generalization, Correlation
Dependency : RelationshipKind
(SCELibrary.RelationshipKinds)
Miscellaneous : RelationshipKind
(SCELibrary.RelationshipKinds)
Reference : RelationshipKind
(SCELibrary.RelationshipKinds)
Generalization : RelationshipKind
(SCELibrary.RelationshipKinds)

Figure 18 - The RelationshipKindSet Metamodel

Generalizations

The *RelationshipKindSet* element inherits the attributes and/or associations of:

- ~~*SCEKindSet*~~*KindSet* (see the section entitled “~~SCEKindSet~~KindSet” for more information).

Further, the ~~*SCEKindSet*~~*KindSet* element inherits the attributes and/or associations of:

- ~~*SCEPackage*~~*Package* (see the section entitled “~~SCEPackage~~Package” for more information).

Further, the ~~*SCEPackage*~~*Package Kind* element inherits the attributes and/or associations of:

Commented [SW190]: This text was updated for the resolution of Issue SCE-90/SCE-95. removing SCE prefix.

- ~~*SCEElement* (see the section entitled "SCEElement" for more information).~~

~~Further, the *SCEElement* element inherits the attributes and/or associations of:~~

- ~~*SCERootElement*~~*BaseElement* (see the section entitled "BaseElement~~BaseElementSCERootElement~~" for more information).

Commented [SW191]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The following table presents the additional attributes and/or associations for *RelationshipKindSet*:

Table 20. RelationshipKindSet Attributes and/or Associations

Property/Association	**Description**
relationshipKind : RelationshipKind [0..*]	The list of terms is a set of *RelationshipKinds* which can be linked to an external ontology through the *Kinds'* `conceptReference` property.

8.6.4 RelationshipKind

Commented [SW192]: This section was moved from 8.4 Internal Relationships to the SCEKindSets section

This class is a type of *Kind* whose instances serve as the `terms` for a ~~*SCEKindSet*~~*KindSet*. A *RelationshipKind* instance is ~~used~~used to specify the kind of relationship that exists between two modeling elements referenced by the *ElementRelationship* and *ElementRelationshipType* elements. Instead of being defined a fixed enumerated list, the kinds can be defined through a class (*RelationshipKind*) and instances of that class (as shown below). The instances defined in the **SCE** Library SHALL be included in any **SCE** implementation. However, the implementation can allow additional instances of the class if required for a particular modeling situation (see the section entitled "RelationshipKinds" for more information).

Commented [SW193]: this text was changed for Issue SCE-7/SCE-8 (Semantic Reference changes)

In practice, when a modeler creates a model with a *ElementRelationship* and *ElementRelationshipType*, the *RelationshipKind* will be instantiated by one of the ~~six~~ seven instances in the Library.

Commented [SW194]: This text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues.

Generalizations

The *RelationshipKind* element inherits the attributes and/or associations of:

- *Kind* (see the section entitled "Kind" for more information).

Further, the *Kind* element inherits the attributes and/or associations of:

Commented [SW195]: Changes to the generalizations chain is part of the resolution for Issue SCE-7/SCE-8 (Semantic Reference changes and Vocabulary changes)

- ~~*SCERootElement*~~*BaseElement* (see the section entitled "BaseElement~~BaseElementSCERootElement~~" for more information).

Commented [SW196]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Properties

The *RelationshipKind* element does not have any additional attributes and/or associations.

Standard Terms KindSet

Commented [SW197]: this text was changed for Issue SCE-7/SCE-8 (Semantic Reference changes)

The following table presents a description for the included instances for *RelationshipKind*:

Table 21. RelationshipKind Instances

Instance	Description
Composition	`Composition` indicates that the `source` element is composed of, in part, the `target` element. Other elements could be included in this composition.
Containment	`Containment` indicates that the `source` element is a container for the `target` element.
Correlation	`Correlation` indicates that the `source` element is correlated with the `target` element. This is often used when a mapping is required between the structures of two data elements.
Dependency	`Dependency` indicates that `target` element is dependent in some way on the `source` element.
Miscellaneous	`Miscellaneous` indicates that `source` element has some relationship with the `target` element that is of a kind that is not expressed through the other *RelationshipKind* instances.
Reference	`Reference` indicates that `source` element references the `target` element.
Generalization	`Generalization` indicates that the `source` element is a generalization of the `target` element (which is based on and extends the `source`).

9 SCE Library

A Library is included in **SCE** to provide standard instances that should be implemented by tools supporting **SCE** through their implementing of a modeling language dependent on **SCE**. Currently, **SCE** defines the instances for one sub-package named *RelationshipKinds* (See next section).

9.1 RelationshipKinds

The *RelationshipKinds* package contains one instance of a~~n~~ *~~SCEKindSet~~KindSet~~SCEKindSetSCEKindSetSCEVocabulary~~*: `RelationshipKinds` which is provided by the **SCE** Library. The purpose of this ~~vocabulary~~ kind set is to provide a set of standard terms, which are instances of the *RelationshipKind* element.

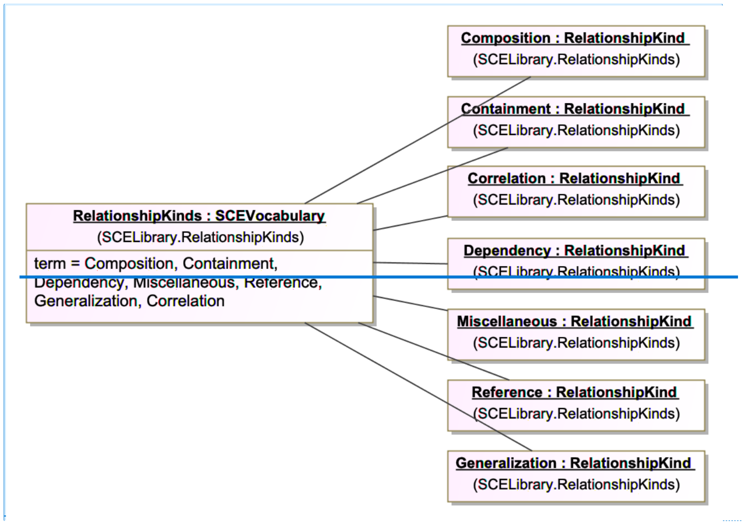
Commented [SW198]: this text was changed for Issue SCE-7/SCE-8 (Semantic Reference changes)

The *RelationshipKind* element is used to specific the kind of relationship that exists between two modeling elements referenced by the *ElementRelationship* and *ElementRelationshipType* elements. Instead of defined a fixed enumerated list, the kinds can be defined through a class (*RelationshipKind*) and instances of that class (as shown below). The instances defined in this Library SHALL be included in any **SCE** implementation. However, the implementation can allow additional instances of the class if required for a particular modeling situation.

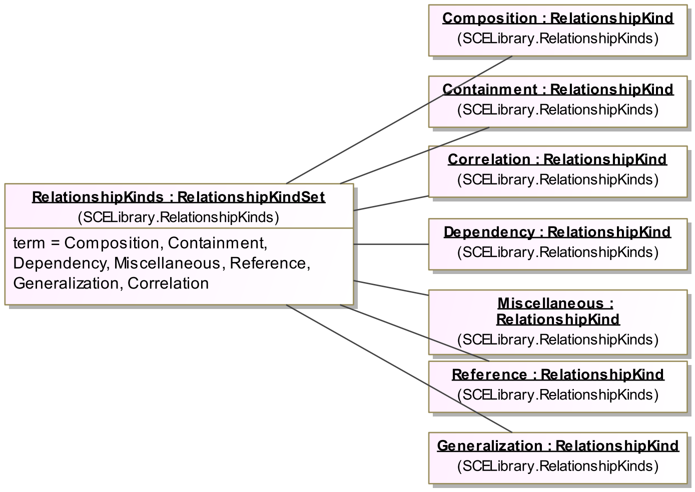
In practice, when a modeler creates a model with a *ElementRelationship* and *ElementRelationshipType*, the *RelationshipKind* will be instantiated by one of the six instances in this Library.

The following figure presents the instances for the *RelationshipKind* element that are `terms` for the instance (`RelationshipKinds`) of the *~~SCEVocabulary~~ RelationshipKindSet* element:

Commented [SW199]: this text was changed for Issue SCE-7/SCE-8 (Semantic Reference changes)


Composition : RelationshipKind
(SCELibrary.RelationshipKinds)
Containment : RelationshipKind
(SCELibrary.RelationshipKinds)
Correlation : RelationshipKind
(SCELibrary.RelationshipKinds)
RelationshipKinds : SCEVocabulary
(SCELibrary.RelationshipKinds)
term = Composition, Containment, Dependency, Miscellaneous, Reference, Generalization, Correlation
Dependency : RelationshipKind
(SCELibrary.RelationshipKinds)
Miscellaneous : RelationshipKind
(SCELibrary.RelationshipKinds)
Reference : RelationshipKind
(SCELibrary.RelationshipKinds)
Generalization : RelationshipKind
(SCELibrary.RelationshipKinds)

Commented [SW200]: this figure was changed for Issue SCE-7/SCE-8 (Semantic Reference changes). RelationshipKinds was reset to be based on RelationshipKindSet (a new class) instead of SCEVocabulary.


Composition : RelationshipKind
(SCELibrary.RelationshipKinds)
Containment : RelationshipKind
(SCELibrary.RelationshipKinds)
Correlation : RelationshipKind
(SCELibrary.RelationshipKinds)
RelationshipKinds : RelationshipKindSet
(SCELibrary.RelationshipKinds)
term = Composition, Containment, Dependency, Miscellaneous, Reference, Generalization, Correlation
Dependency : RelationshipKind
(SCELibrary.RelationshipKinds)
Miscellaneous : RelationshipKind
(SCELibrary.RelationshipKinds)
Reference : RelationshipKind
(SCELibrary.RelationshipKinds)
Generalization : RelationshipKind
(SCELibrary.RelationshipKinds)

~~Figure 24 -~~ Figure 19 - The RelationshipKinds Instance Model

The following table presents a description for the included instances for *RelationshipKind*:

~~Table 25.~~Table 22. RelationshipKind Instances

Instance	Description
Composition	`Composition` indicates that the `source` element is composed of, in part, the `target` element. Other elements could be included in this composition.

Containment	Containment indicates that the source element is a container for the target element.
Correlation	Correlation indicates that the source element is correlated with the target element. This is often used when a mapping is required between the structures of two data elements.
Dependency	Dependency indicates that target element is dependent in some way on the source element.
Miscellaneous	Miscellaneous indicates that source element has some relationship with the target element that is of a kind that is not expressed through the other *RelationshipKind* instances.
Reference	Reference indicates that source element references the target element.
Generalization	Generalization indicates that the source element is a generalization of the target element (which is based on and extends the source).

10 Exchange Formats

In general, **SCE** models will not be interchanged independently, but will be interchanged in the context of another modeling specification, such as ~~**BKPMN,**~~ **SDMN**~~**, or PPMN**~~. Thus, this section specifies characteristics of exchanging **SCE** models.

Commented [SW201]: This text was updated for the resolution of Issue SCE-119/SCE-120. removing non-existing specifications.

10.1 Interchanging Incomplete Models

In practice, it is common for models to be interchanged before they are complete. This occurs frequently when doing iterative modeling, where one user (such as a subject matter expert or business person) first defines a high-level model, and then passes it on to another user to be completed and refined.

Such “incomplete” models are ones in which all of the mandatory attributes have not yet been filled in, or the cardinality lowerbound of attributes and associations has not been satisfied.

XMI allows for the interchange of such incomplete models. With **SCE**, we extend this capability to interchange of XML files based on the **SCE XSD**. In such XML files, implementers are expected to support this interchange by:

- Disregarding missing attributes that are marked as ‘required’ in the XSD.
- Reducing the lower bound of elements with ‘minOccurs’ greater than 0.

10.2 XSD

10.2.1 Document Structure

A domain-specific set of model elements is interchanged in one or more **SCE** files. The root element of each file SHALL be an instance sub-class of *Model*. ~~<SCE:SCEDefinitions>.~~ The set of files SHALL be self-contained, i.e., all definitions that are used in a file SHALL be imported directly or indirectly using the <~~SCE~~sce:i~~I~~mport:Import> element.

Each file SHALL declare a “targetN~~n~~amespacenamespace” that MAY differ between multiple files of one model.

Commented [SW202]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW203]: This text was updated for the resolution of Issue SCE-69/SCE-106. Editorial issues.

Commented [SW204]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

SCE files MAY import non-**SCE** files (such as XSDs) if the contained elements use external definitions.

The XML namespace URIs for SCE 1.0 and backwards-compatible 1.x versions of SCE are fixed at:

- https://www.omg.org/spec/SCE/
- https://www.omg.org/spec/SCE/SCEDI/
- https://www.omg.org/spec/SCE/DI/
- https://www.omg.org/spec/SCE/DC/

In addition, the root element of SCE-based XML files MUST include an xsi:schemaLocation attribute that points to the concrete schema files of the versions of all namespaces, i.e. the URLs of the XSDs that are publicly hosted by OMG with dated version stamps, e.g.

```
<?xml version="1.0" encoding="UTF-8"?>
<sdmn:sharedDataModel id="HelloWorldDataModel"
    targetNamespace="https://example.org/hello-world/shared-data-model"
    xmlns="https://example.org/hello-world/shared-data-model"
    xmlns:sdmn="https://www.omg.org/spec/SDMN/"
    xmlns:sce="https://www.omg.org/spec/SCE/"
    xmlns:scedi="https://www.omg.org/spec/SCE/SCEDI/"
    xmlns:di="https://www.omg.org/spec/SCE/DI/"
    xmlns:dc="https://www.omg.org/spec/SCE/DC/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      https://www.omg.org/spec/SDMN/
      https://www.omg.org/spec/SDMN/20240210/SDMN.xsd
      https://www.omg.org/spec/SCE/
      https://www.omg.org/spec/SCE/20240210/SCE.xsd
      https://www.omg.org/spec/SCE/SCEDI/
      https://www.omg.org/spec/SCE/20240210/SCEDI.xsd
      https://www.omg.org/spec/SCE/DI/
      https://www.omg.org/spec/SCE/20240210/DI.xsd
      https://www.omg.org/spec/SCE/DC/
      https://www.omg.org/spec/SCE/20240210/DC.xsd
    ">
  <sce:import name="Hello_World_Item_Definitions" location="hello-world-item-
definitions.sdmn"
    importType="https://www.omg.org/spec/SDMN/"
    namespace="https://example.org/hello-world/item-definitions"/>
  <!-- ... -->
</sdmn:sharedDataModel>
```

When importing models, tools MUST read the `xsi:schemaLocation` attribute to identify which exact versions of SCE and SCE-based languages are used in the file based on the schema URLs associated with each namespace URI. If the `xsi:schemaLocation` indicates that a newer version than the one supported by the importing tool is used in the XML file, the tool MUST report a warning to the user. Furthermore, an XML schema validation with the schema version supported by the importing tool can reveal whether or not newer language features are used in the XML file.

Unless defined otherwise, the requirement to set xsi:schemaLocation and the option to read it as a version identifier are inherited by SCE-based languages, e.g. SDMN.

Commented [SAW205]: This text was updated for the resolution of Issue SCE-117/SCE-118. Use fixed Namespace URI and xsi:schemaLocation

10.2.2 References within the SCE XSD

Many **SCE** elements that may need to be referenced contain IDs and within the **SCE** XSD, references to elements are expressed via these IDs. The XSD IDREF type is the traditional mechanism for referencing by IDs, however it can only reference an element within the same file. **SCE** elements of type *~~SCERootElement~~BaseElement* support referencing by ID, across files, by utilizing ~~an href attribute whose value must be a valid URI reference [RFC 3986]~~QNames. A QName consists of two parts: an optional namespace prefix and a local part~~ where the path components may be absolute or relative, the reference has no query component, and the fragment consists of the~~ .

Commented [SW206]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

When used to reference a **SCE** element, the local part is expected to be the ~~value of the~~ id of the referenced **SCE** ~~**SCE**~~ element.

Commented [SW207]: This text was updated for the resolution of Issue SCE-69/SCE-106. Editorial issues

11 SCE Diagram Interchange (SCE DI)

11.1 Scope

This chapter specifies the meta-model and schema for **SCE** ~~1.0~~ Diagram Interchange (**SCE DI**). The **SCE DI** is meant to facilitate the interchange of **SCE**-dependent diagrams between tools rather than being used for internal diagram representation by the tools. The simplest interchange approach to ensure the unambiguous rendering of a **SCE**-dependent diagram was chosen for **SCE DI**. As such, **SCE DI** does not aim to preserve or interchange any "tool smarts" between the source and target tools (e.g., layout smarts, efficient styling, etc.).

Commented [SW208]: This text was updated for the resolution of Issue SCE-69/SCE-106. Editorial issues

SCE DI does not ascertain that the **SCE**-dependent diagram is syntactically or semantically correct. This version of **SCE DI** focuses on the interchange of *DiagramArtifacts* that can be used in any modeling language that is dependent on **SCE**.

11.2 Diagram Definition and Interchange

The **SCE DI** metamodel, similar to the **SCE** abstract syntax meta-model, is defined as a MOF-based meta-model. As such, its instances can be serialized and interchanged using XMI. **SCE DI** is also defined by an XML schema. Thus, its instances can also be serialized and interchanged using XML.

The **SCE DI** metamodel and schema are harmonized with the OMG Diagram Definition (**DD**) standard version 1.1. The referenced **DD** contains two main parts: the Diagram Commons (**DC**) and the Diagram Interchange (**DI**). The **DC** defines common types like bounds and points, while the **DI** provides a framework for defining domain-specific diagram models. As a domain-specific **DI**, **SCE DI** defines a few new meta-model classes that derive from the abstract classes from **DI**.

The focus of **SCE DI** is the interchange of laid out shapes and edges that constitute a **SCE**-dependent diagram. Each shape and edge references a particular **SCE** model element. The referenced **SCE** model elements are all part of the actual **SCE** model. As such, **SCE DI** is meant to only contain information that is neither present nor derivable, from the **SCE** model whenever possible. Simply put, to render a **SCE**-dependent diagram both the **SCE DI** instance(s) and the referenced **SCE** model are REQUIRED.

From the **SCE DI** perspective, a **SCE**-dependent diagram is a particular snapshot of a **SCE** model at a certain point in time. Multiple **SCE**-dependent diagrams can be exchanged referencing model elements from the same **SCE** model. Each diagram may provide an incomplete or partial depiction of the content of the **SCE** model. As described in clause 10 ~~clause 12~~, a **SCE** model package consists of one or more files. Each file may contain any number of **SCE**-dependent diagrams. The exporting tool is free to decide how many diagrams are exported and the importing tool is free to decide if and how to present the contained diagrams to the user.

Commented [SW209]: This text was updated for the resolution of Issue SCE-69/SCE-106. Editorial issues

11.3 SCE Diagram Interchange Meta-Model

11.3.1 How to read this chapter

Clause 11.3.4~~10.4~~ describes in detail the meta-model used to keep the layout and the look of **SCE**-dependent Diagrams. Clause 11.4~~10.5~~ presents in tables a library of the **SCE** element depictions and an unambiguous resolution between a referenced **SCE** model element and its depiction.

Commented [SW210]: This text was updated as part of the resolution to Issue SCE-14/SCE-72. Fixing the cross references.

11.3.2 Overview

The **SCE DI** is an instance of the OMG **DI** meta-model. The basic concept of **SCE DI**, as with diagram interchange in general, is that serializing a diagram [~~*SCEDi*~~*Diagram*~~*SCEDiagram*~~] for interchange requires the specification of a collection of shapes [~~*SCEShape*~~*Shape*] and edges [~~*SCEEdge*~~*Edge*].

The **SCE DI** classes only define the visual properties used for depiction. All other properties that are REQUIRED for the unambiguous depiction of the **SCE** element are derived from the referenced **SCE** element

[~~SCEElement~~ElementRef~~SCEElementRef~~].

SCE-dependent diagrams may be an incomplete or partial depiction of the content of the **SCE** model. Some **SCE** elements from a **SCE** model may not be present in any of the diagram instances being interchanged.

SCE DI does not directly provide for any containment concept. The *~~SCEDiagram~~Diagram* is an ordered collection of mixed *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s). The order of the *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s) inside a *~~SCEDiagram~~Diagram* determines their Z-order (i.e., what is in front of what). *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s) that are meant to be depicted "on top" of other *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s) MUST appear after them in the *~~SCEDiagram~~Diagram*. Thus, the exporting tool MUST order all *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s) such that the desired depiction can be rendered.

Commented [SAW211]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

11.3.3 Measurement Unit

As per OMG DD, all coordinates and lengths defined by **SCEDI** are assumed to be in user units, except when specified otherwise. A user unit is a value in the user coordinate system, which initially (before any transformation is applied) aligns with the device's coordinate system (for example, a pixel grid of a display). A user unit, therefore, represents a logical rather than physical measurement unit. Since some applications might specify a physical dimension for a diagram as well (mainly for printing purposes), a mapping from a user unit to a physical unit can be specified as a diagram's resolution. Inch is chosen in this specification to avoid variability, but tools can easily convert from/to other preferred physical units. Resolution specifies how many user units fit within one physical unit (for example, a resolution of 300 specifies that 300 user units fit within 1 inch on the device).

11.3.4 Elements

The following sections define the elements necessary for exchanging the diagrams from BPM+ modeling languages that are dependent on **SCE**. Specifically, the graphical *DiagramArtifacts* that may be used in the diagram.

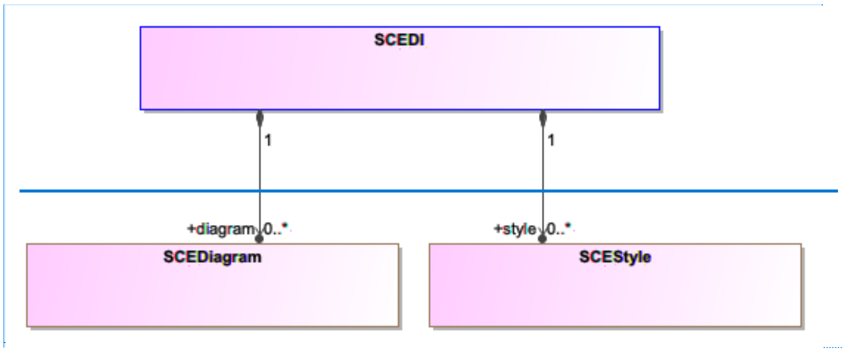
11.3.4.1 ~~SCEDI~~Diagrams

The class *~~SCEDiagram~~Diagrams~~SCEDI~~* is a container for the shared *~~SCEStyle~~Style* and all the *~~SCEDiagram~~Diagram* defined in a **SCE**-dependent modeling package.

Commented [SW212]: This text was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

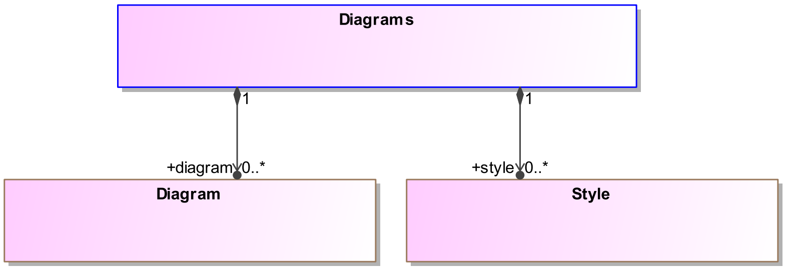
Commented [SAW213]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

The following figure shows the *SCEDI* metamodel diagram.



Commented [SW214]: This figure was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use. SCEDI was renamed to SCEDiagrams

Commented [SW215]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.



~~Figure 25 -~~ Figure 20 - The ~~SCEDI~~ Diagrams Metamodel

Generalizations

The *~~SCEDiagram~~Diagrams~~SCEDI~~* element does not inherit any attributes or associations of from another element.

Properties

The following table presents the additional attributes and/or associations for *~~SCEDI~~Diagrams*:

~~Table 26.~~Table 23. ~~SCEDiagram~~Diagrams ~~SCEDI~~ Attributes and/or Associations

Property/Association	Description
diagram : ~~SCEDiagram~~Diagram [0..*]	A list of *~~SCEDiagram~~Diagrams~~SCEDiagrams~~*.
style : ~~SCEStyle~~Style [0..*]	A list of shared ~~SCEStyle~~Stylethat~~SCEStylethat~~ can be referenced by all **SCE**-dependent diagrams and *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~*.

11.3.4.2 ~~SCEDiagram~~Diagram

The ~~abstract~~ class *~~SCEDiagram~~Diagram* specializes DI::Diagram. It is a kind of Diagram that represents a depiction of all or part of a **SCE**-dependent model. It is contained within the *SCEDI* element (see above). The languages that are dependent on **SCE** will this~~define concrete diagrams based on *SCEDiagram*~~this class..

~~SCEDiagram~~Diagram is the container of *~~SCEDiagram~~DiagramElement (~~SCEShape~~Shape~~SCEDiagramElement (SCEShape~~*(s) and *~~SCEEdge~~Edge*(s)). *~~SCEDiagram~~Diagram* cannot include other *~~SCEDiagram~~Diagrams~~SCEDiagrams~~*.

A *~~SCEDiagram~~Diagram* can define a *~~SCEStyle~~Style* locally and/or it can refer to a shared one defined in the **SCEDI**. Properties defined in the local style overrides the one in the referenced shared style. That combined style (shared and local) is the default style for all the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* contained in this *~~SCEDiagram~~Diagram*.

The *~~SCEDiagram~~Diagram* class represents a two-dimensional surface with an origin of (0, 0) at the top left corner. This means that the x and y axes have increasing coordinates to the right and bottom. Only positive coordinates are allowed for diagram elements that are nested in a *~~SCEDiagram~~Diagram*.

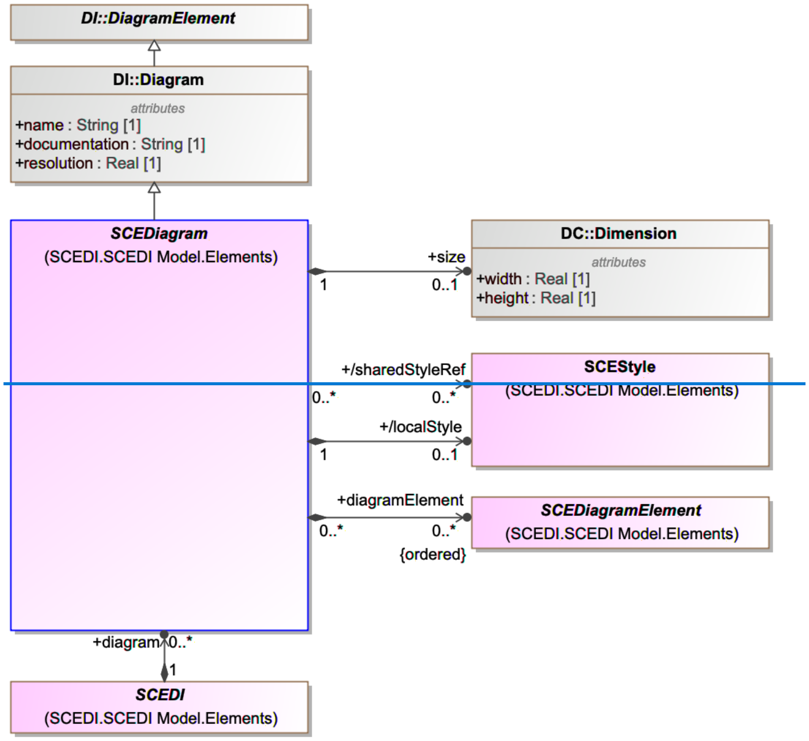
The following figure shows the *~~SCEDiagram~~Diagram* metamodel diagram.

Commented [SAW217]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

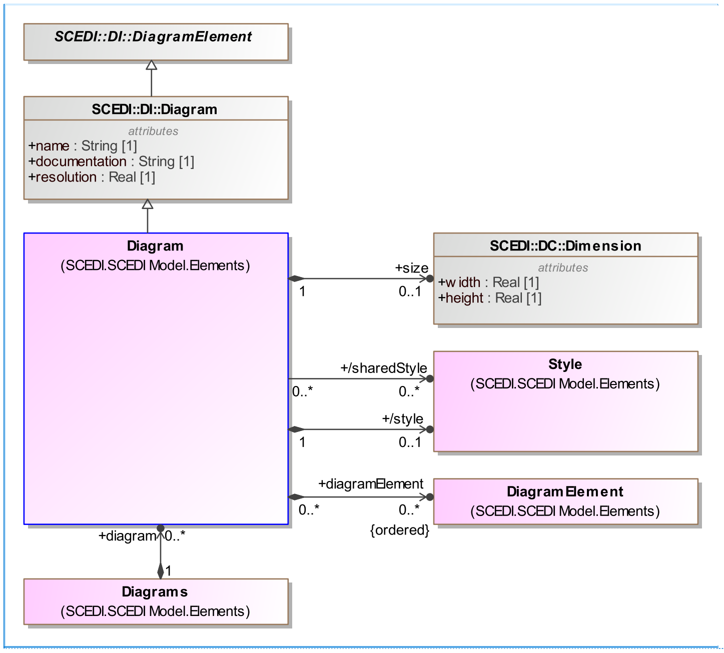
Commented [SW216]: This text was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

Commented [SW218]: This text was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

Commented [SAW219]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.



Commented [SW220]: This figure was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use. SCEDiagram was made concrete.



Commented [SW221]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW222]: This figure was also updated for the resolution of Issue SCE-115/SCE-116. Fix Style association role names.

~~Figure 26 -~~ Figure 21 - The ~~SCEDiagram~~Diagram Metamodel

Generalizations

The *~~SCEDiagram~~Diagram* element inherits the attributes and/or associations of:

Commented [SAW226]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

- *Diagram* (see the section entitled "Diagram" for more information).

Further, the *Diagram* element inherits the attributes and/or associations of:

- *DiagramElement* (see the section entitled "~~DiagramElement~~DiagramElement" for more information).

Properties

The following table presents the additional attributes and/or associations for *~~SCEDiagram~~Diagram*:

~~Table 27.~~Table 24. ~~SCEDiagram~~Diagram Attributes and/or Associations

Property/Association	Description
diagramElement : ~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~ [0..*]	A list of *~~SCEDiagram~~DiagramElements* *(~~SCEShape~~Shape~~SCEDiagramElements (SCEShape~~* and *~~SCEEdge~~Edge*) that are depicted in the **SCE**-dependent diagram.
~~**diagramRef** : SCEDiagram [1]~~	~~The diagram that the DI is representing.~~
~~**localStyle**~~ **style** : ~~SCEStyle~~Style [0..1]	A *~~SCEStyle~~Style* that defines the default styling for this diagram. Properties defined in that style override the ones in the `sharedStyle`.

Commented [SW223]: This row was removed by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

Commented [SW224]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo Style association names.

sharedStyle~~Ref~~ : ~~SCEStyle~~Style [0..*]	A reference to a *~~SCEStyle~~Style* defined in the SCEDI that serves as the default styling of the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* in the **SCE**-dependent diagram.
size : DC:Dimension [0..1]	The size of this diagram. If not specified, the the **SCE**-dependent diagram is unbounded.

Commented [SW225]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo Style association names.

11.3.4.3 ~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~

The *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* class is contained by the *~~SCEDiagram~~Diagram* and is the base class for *~~SCEShape~~Shape* and *~~SCEEdge~~Edge*.

~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~ inherits its styling from its parent *~~SCEDiagram~~Diagram*. In addition, it can refer to one of the shared *~~SCEStyle~~Style* defined in the **SCEDI** and/or it can define a local style. See section below for more details on styling.

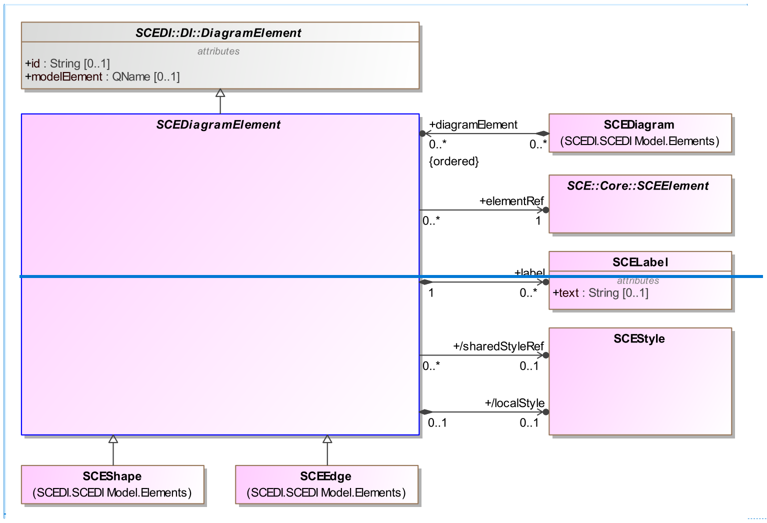
~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~ MAY also contain a *~~SCE~~Label* when it has a visible text label. If no *~~SCE~~Label* is defined, the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* should be depicted without a label.

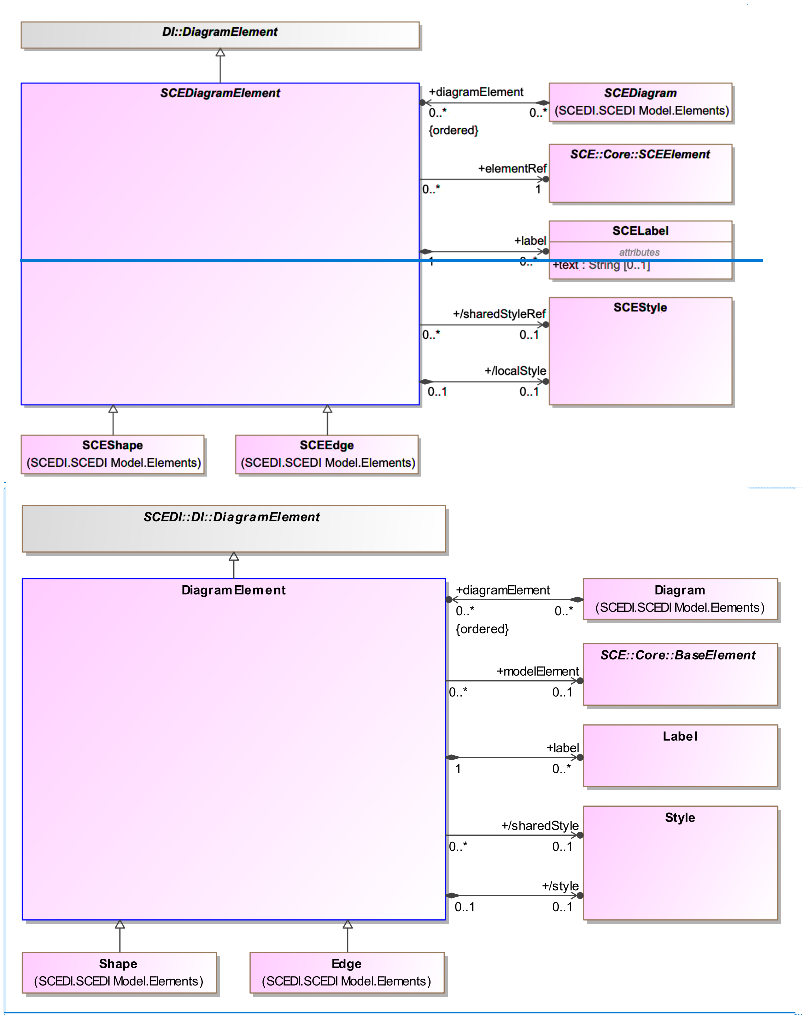
The following figure shows the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* metamodel diagram.

Commented [SW227]: This figure was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use. SCEDiagramElement was made concrete.

Commented [SAW228]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW229]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.





~~Figure 27 -~~ Figure 22 - The ~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~ Metamodel

Generalizations

The *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* element inherits the attributes and/or associations of:

- *DiagramElement* (see the section entitled "~~DiagramElement~~DiagramElement" for more information).

Properties

The following table presents the additional attributes and/or associations for *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~*:

Commented [SW230]: This figure was also updated for the resolution of Issue SCE-111/SCE-112. making the modelElement attribute optional.

Commented [SW231]: This figure was also updated for the resolution of Issue SCE-113/SCE-114. removing text attribute from Label.

Commented [SAW237]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

~~Table 28.~~Table 25. ~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~ Attributes and/or Associations

Property/Association	Description
label : ~~SCE~~Label [0..*]	An optional label when the **SCE**-dependent Element has a visible text label.
~~**localStyle**~~ **style** : ~~SCEStyle~~Style [0..1]	A *~~SCEStyle~~Style* that defines the styling for this element.
modelElement : ~~SCEElement~~BaseElement [0..1]~~**sceEsceElementRef** : SCEElement [1]~~	A reference to the concrete instance of the *~~SCEElement~~BaseElement* that is being depicted.
sharedStyle~~Ref~~ : ~~SCEStyle~~Style [0..1]	A reference to a *~~SCEStyle~~Style* defined in the *SCEDI*.

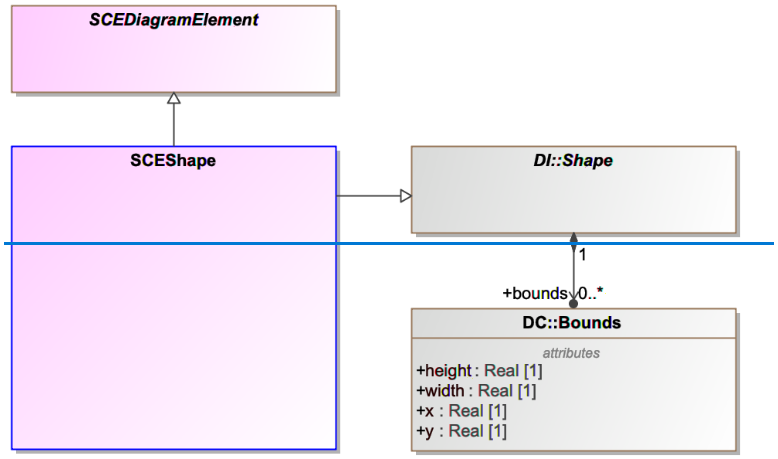
11.3.4.4 ~~SCEShape~~Shape

The *~~SCEShape~~Shape* class specializes DI::Shape and *~~SCEDiagram~~DiagramElement.~~SCEDiagramElement.~~* It is a kind of Shape that depicts an *~~SCEElement~~Element* from the **SCE**-dependent model.

~~SCEShape~~Shape represents a **Group** or a **Text Annotation** that is depicted on the diagram. **SCE**-dependent models may add additional shapes to their diagrams.

~~SCEShape~~Shape has no additional properties but a **SCE**-dependent model may extend this class to add properties that are used to further specify the appearance of some shapes that cannot be deduced from the **SCE**-dependent model.

The following figure shows the *~~SCEShape~~Shape* metamodel diagram.



Commented [SW232]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo Style association names.

Commented [SAW235]: This text was updated from the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. merging RootElement and Element.

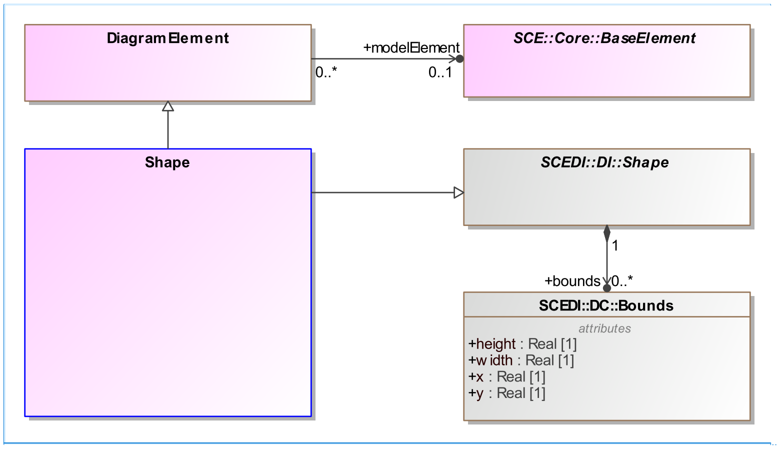
Commented [SAW233]: This text was updated from the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. merging RootElement and Element.

Commented [SW234]: This text was updated for the resolution of Issue SCE-111/SCE-112. making the modelElement attribute optional.

Commented [SW236]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo Style association names.

Commented [SAW238]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the “SCE” prefix for model elements.

Commented [SW239]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the “SCE” prefix for model elements.



~~Figure 28 -~~ Figure 23 - The ~~SCEShape~~Shape Metamodel

Commented [SW240]: This figure was also updated for the resolution of Issue SCE-111/SCE-112. making the modelElement attribute optional.

Generalizations

The *~~SCEShape~~Shape* element inherits the attributes and/or associations of:

- *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* (see the section entitled "~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~SCEDiagramElement" for more information).

Further, the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* element inherits the attributes and/or associations of:

- *DiagramElement* (see the section entitled "~~DiagramElement~~DiagramElement" for more information).

In addition, the *~~SCEShape~~Shape* element inherits the attributes and/or associations of:

- *Shape* (see the section entitled "Shape" for more information).

Properties

The *~~SCEShape~~Shape* element does not have any additional attributes and/or associations.

11.3.4.5 ~~SCEEdge~~Edge

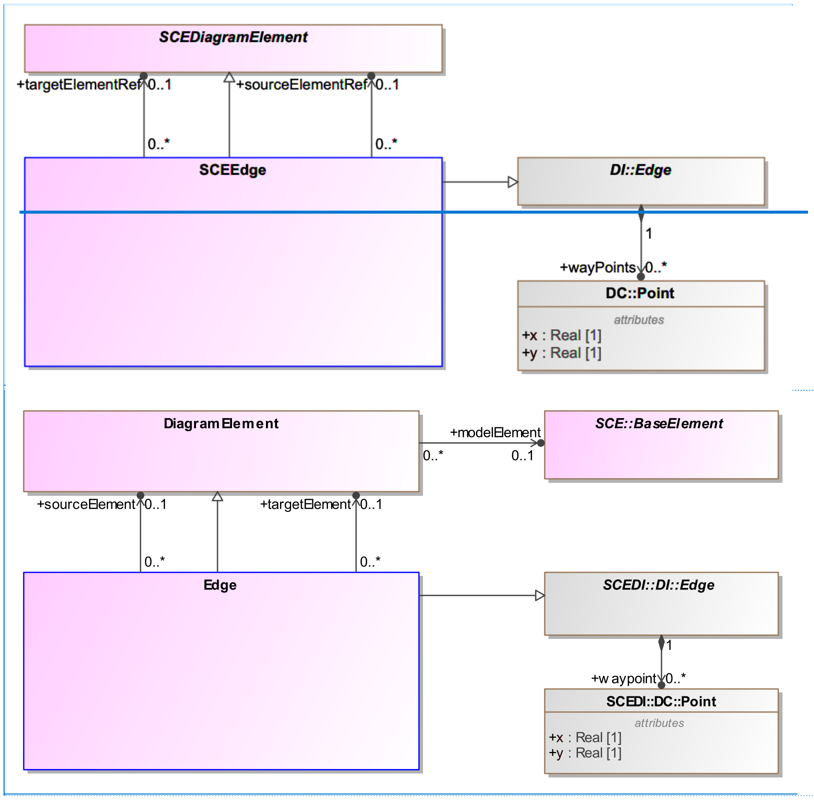
The *~~SCEEdge~~Edge* class specializes DI::Edge and *~~SCEDiagram~~DiagramElement.~~SCEDiagramElement.~~* It is a kind of Edge that can depict a relationship between two **SCE**-dependent model elements.

~~SCEEdge~~Edge ~~are~~ is used to depict **Associations** in the **SCE**-dependent model. Since *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* might be depicted more than once, `sourceElement` and `targetElement` attributes allow to determine to which depiction an *~~SCEEdge~~Edge* is connected. When *~~SCEEdge~~Edge* has a source, its `sourceModelElement` MUST refer to the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* it starts from. That *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* MUST resolved to the *~~SCEElement~~Element* that is the actual source of the **Association**. When it has a target, its targetModelElement MUST refer to the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* where it ends. That *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* MUST resolved to the *~~SCEElement~~BaseElement* that is the actual target of the **Association**.

The following figure shows the *~~SCEEdge~~Edge* metamodel diagram.

Commented [SAW241]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. RootElement and Element were merged and named BaseElement.

Commented [SAW242]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.



~~Figure 29 -~~ Figure 24 - The ~~SCEEdge~~Edge Metamodel

Commented [SW243]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW244]: This figure was also updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

Commented [SW245]: This figure was also updated for the resolution of Issue SCE-111/SCE-112. making the modelElement attribute optional.

Generalizations

The *~~SCEEdge~~Edge* element inherits the attributes and/or associations of:

- *Edge* (see the section entitled "Edge" for more information).

In addition, the *~~SCEEdge~~Edge* element inherits the attributes and/or associations of:

- *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* (see the section entitled "~~SCEDiagramE~~DiagramElement~~SCEDiagramElement~~SCEDiagramElement" for more information).

Further, the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* element inherits the attributes and/or associations of:

- *DiagramElement* (see the section entitled "~~DiagramElement~~DiagramElement" for more information).

Properties

The following table presents the additional attributes and/or associations for *~~SCEEdge~~Edge*:

~~Table 29.~~Table 26. ~~SCEEdge~~Edge Attributes and/or Associations

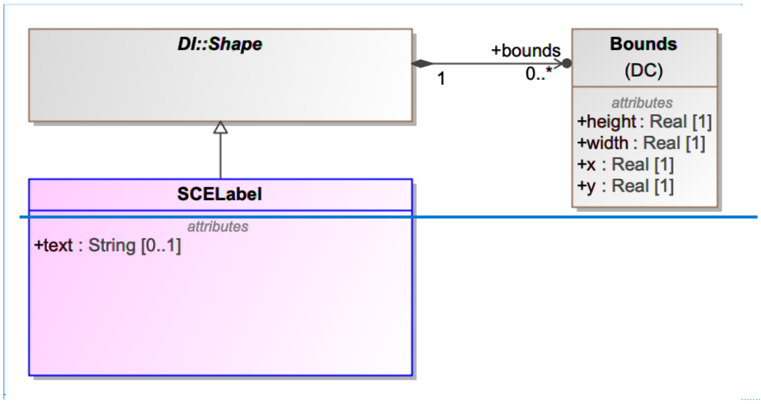
Property/Association	Description
sourceElement : ~~SCEDiagram~~DiagramElement~~**sourceElementRef** : SCEDiagramElement~~ [0..1]	~~The actual *SCEDiagramElement* this *SCEEdge* is connecting from. This MUST be specified when the *SCEEdge* has a source.~~An optional reference to the DiagramElement that this Edge starts from. This attribute MUST ONLY be present if the Edge is depicted starting from a different source than the one referenced by the modelElement of the Edge (from DI:Edge).
targetElement : ~~SCEDiagram~~DiagramElement~~**targetElementRef** : SCEDiagramElement~~ [0..1]	An optional reference to the DiagramElement that this Edge starts from. This attribute MUST ONLY be present if the Edge is depicted ending at a different target than the one referenced by the modelElement of the Edge (from DI:Edge).~~The actual *SCEDiagramElement* this *SCEEdge* is connecting to. This MUST be specified when the *SCEEdge* has a target.~~

11.3.4.6 ~~SCE~~Label

~~*SCE*~~*Label* represents the depiction of some textual information about an element.

A ~~*SCE*~~*Label* is not a top-level element but is always nested inside either a ~~*SCEShape*~~*Shape* or an ~~*SCEEdge*~~*Edge*. It does not have its own reference to a **SCE** element but rather inherits that reference from its parent ~~*SCEShape*~~*Shape* ~~or DMNEdge~~. The textual information depicted by the label is derived from the name attribute of the referenced ~~*SCEElement*~~*BaseElement*.

The following figure shows the ~~*SCE*~~*Label* metamodel diagram.



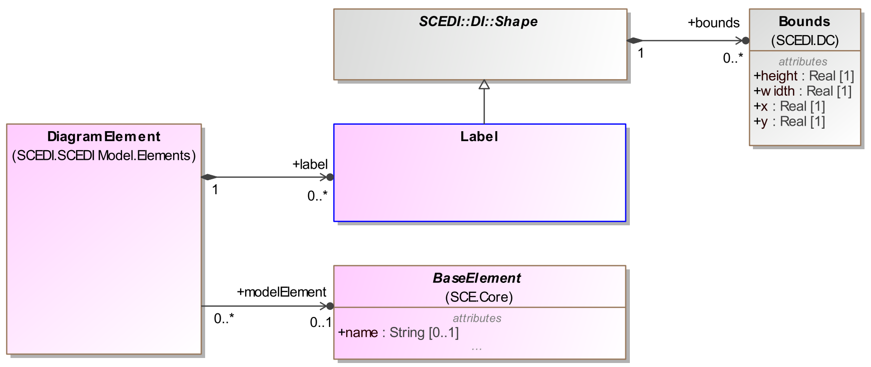
Commented [SAW246]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SAW247]: This text was updated by the resolution of Issue SCE-53/SCE-84. Adjusting SCEDI for direct re-use.

Commented [SAW248]: This text was updated from the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility. merging RootElement and Element.

Commented [SAW249]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW250]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.



~~Figure 30 -~~ Figure 25 - The ~~SCE~~Label Metamodel

Commented [SW251]: This figure was also updated for the resolution of Issue SCE-113/SCE-114. removing text attribute from Label.

Generalizations

The ~~*SCE*~~*Label* element inherits the attributes and/or associations of:

Commented [SAW252]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

- *Shape* (see the section entitled "Shape" for more information).

Properties

The *Label* element does not have any additional attributes and/or associations.

Commented [SW253]: This text was updated for the resolution of Issue SCE-113/SCE-114. removing text attribute from Label.

~~The following table presents the additional attributes and/or associations for *SCELabel*:~~

~~Table 30. SCELabel Attributes and/or Associations~~

~~**Property/Association**~~	~~**Description**~~
~~**text** : String [0..1]~~	~~An optional pretty printed text that MUST be displayed instead of the *SCEElement*SCEElement's name if it is present.~~

11.3.4.7 ~~SCEStyle~~Style

~~*SCEStyle*~~*Style* specializes DC::Style. It is a kind of style that provides appearance options for a ~~*SCEDiagram*~~*DiagramElement*~~*SCEDiagramElement*~~.

~~*SCEStyle*~~*Style* is used to keep some non-normative visual attributes such as colors and font. **SCE** doesn't give any semantic to color and font styling, but tools can decide to use them and interchange them.

~~*SCEDiagram*~~*DiagramElement*~~*SCEDiagramElement*~~ style is calculated by percolating up ~~*SCEStyle*~~*Style* attributes defined at a different level of the hierarchy. Each attribute is considered independently (meaning that a ~~*SCEStyle*~~*Style* attribute can be individually overloaded). The precedence rules are as follow:

- The ~~*SCEStyle*~~*Style* defined by the ~~localS~~style attribute of the ~~*SCEDiagram*~~*DiagramElement*~~*SCEDiagramElement*~~
- The ~~*SCEStyle*~~*Style* referenced by the sharedStyle ~~sharedStyle~~ attribute of the ~~*SCEDiagram*~~*DiagramElement*~~*SCEDiagramElement*~~
- The ~~*SCEStyle*~~*Style* defined by the ~~localS~~style ~~tyle~~ attribute of the parent ~~*SCEDiagram*~~*Diagram*

Commented [SW254]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo Style association names.

Commented [SW255]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo Style association names.

Commented [SW256]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo Style association names.

- The *~~SCEStyle~~Style* referenced by the `sharedStyle` ~~sharedStyle~~ attribute of the parent *~~SCEDiagram~~Diagram*

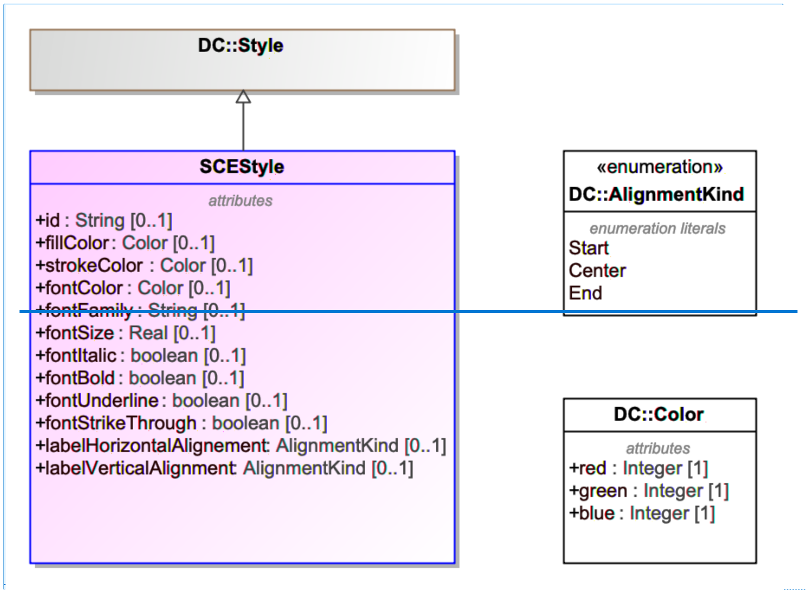
The default attribute value defined in *~~SCEStyle~~Style* attributes.

For example, let's say we have the following:

- *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* has a local *~~SCEStyle~~Style* that specifies the `fillColor` and `strokeColor`
- Its parent *~~SCEDiagram~~Diagram* defines a local *~~SCEStyle~~Style* that specifies the `fillColor` and `fontColor`

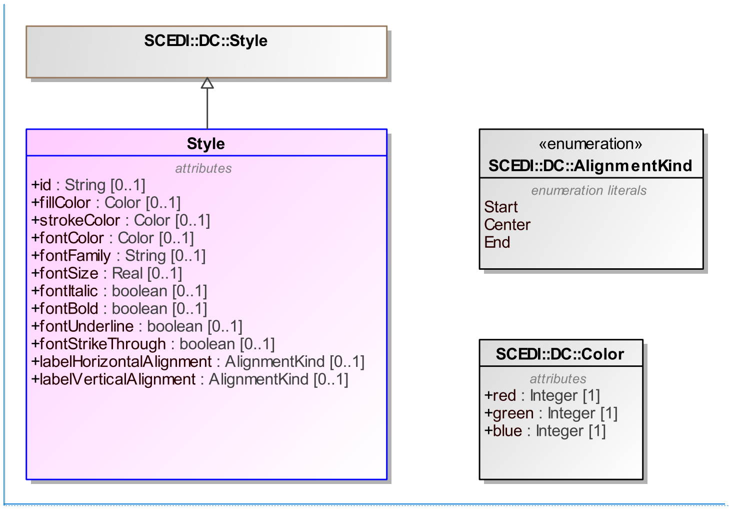
Then the resulting *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* should use:

- The `fillColor` and `strokeColor` defined at the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* level (as they are defined locally).
- The `fontColor` defined at the *~~SCEDiagram~~Diagram* level (as the `fillColor` was overloaded locally).
- All other *~~SCEStyle~~Style* attributes would have their default values.


DC::Style
SCEStyle
attributes
+id : String [0..1]
+fillColor : Color [0..1]
+strokeColor : Color [0..1]
+fontColor : Color [0..1]
~~+fontFamily : String [0..1]~~
+fontSize : Real [0..1]
+fontItalic : boolean [0..1]
+fontBold : boolean [0..1]
+fontUnderline : boolean [0..1]
+fontStrikeThrough : boolean [0..1]
+labelHorizontalAlignement: AlignmentKind [0..1]
+labelVerticalAlignment: AlignmentKind [0..1]
«enumeration»
DC::AlignmentKind
enumeration literals
Start
Center
End
DC::Color
attributes
+red : Integer [1]
+green : Integer [1]
+blue : Integer [1]

Commented [SAW257]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Commented [SW258]: This figure was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.


SCEDI::DC::Style
Style
attributes
+id : String [0..1]
+fillColor : Color [0..1]
+strokeColor : Color [0..1]
+fontColor : Color [0..1]
+fontFamily : String [0..1]
+fontSize : Real [0..1]
+fontItalic : boolean [0..1]
+fontBold : boolean [0..1]
+fontUnderline : boolean [0..1]
+fontStrikeThrough : boolean [0..1]
+labelHorizontalAlignment : AlignmentKind [0..1]
+labelVerticalAlignment : AlignmentKind [0..1]
«enumeration»
SCEDI::DC::AlignmentKind
enumeration literals
Start
Center
End
SCEDI::DC::Color
attributes
+red : Integer [1]
+green : Integer [1]
+blue : Integer [1]

Commented [SW259]: This figure was also updated for the resolution of Issue SCE-115/SCE-116. Fix typo in labelHorizontalAlignment property name.

~~Figure 31 -~~Figure 26 - The ~~SCEStyle~~Style Metamodel

Commented [SAW261]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Generalizations

The *~~SCEStyle~~Style* element inherits the attributes and/or associations of:

- *Style* (see the section entitled "Style" for more information).

Properties

The following table presents the additional attributes and/or associations for *~~SCEStyle~~Style*:

~~Table 31.~~Table 27. ~~SCEStyle~~Style Attributes and/or Associations

Property/Association	Description
fillColor : Color [0..1]	The color use to fill the shape. Doesn't apply to *~~SCEEdge~~Edge*. The default is `white`.
fontBold : boolean [0..1]	If the text should be displayed in Bold. The default is `false`.
fontColor : Color [0..1]	The color use to write the label. The default is `black`.
fontFamily : String [0..1]	A comma-separated list of Font Name that can be used to display the text. The default is Arial.
fontItalic : boolean [0..1]	If the text should be displayed in Italic. The default is `false`.
fontSize : Real [0..1]	The size in points of the font to use to display the text. The default is 8.
fontStrikeThrough : boolean [0..1]	If the text should be stroke through. The default is `false`.
fontUnderline : boolean [0..1]	If the text should be underlined. The default is `false`.

id : String [0..1]	A unique id for this style so it can be referenced. Only styles defined in the **SCEDI** can be referenced by *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* and *~~SCEDiagram~~Diagram*.
labelHorizontalAlignement : AlignmentKind [0..1]	How text should be positioned horizontally within the Label bounds. Default depends of the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* the label is attached to (see section below).
labelVerticalAlignment : AlignmentKind [0..1]	How the text should be positioned vertically inside the Label bounds. Default depends of the *~~SCEDiagram~~DiagramElement~~SCEDiagramElement~~* the label is attached to (see section below). Start means “top” and end means “bottom”.
strokeColor : Color [0..1]	The color use to draw the shape borders. The default is `black`.

11.4 Notation

As a specification that contains notation, **SCE** specifies the depiction for **SCE** *DiagramArtifact* elements.

Serializing a **SCE** diagram for interchange requires the specification of a collection of *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s) in the *~~SCEDiagram~~Diagram* (see sections above). The *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s) attributes must be populated in such a way as to allow the unambiguous rendering of the **SCE**-dependent diagram by the receiving party. More specifically, the *~~SCEShape~~Shape*(s) and *~~SCEEdge~~Edge*(s) MUST reference **SCE** model elements. If no *~~SCEElement~~Element* is referenced or if the reference is invalid, it is expected that this shape or edge should not be depicted.

When rendering a **SCE**-dependent diagram, the correct depiction of a *~~SCEShape~~Shape* or *~~SCEEdge~~Edge* depends mainly on the referenced **SCE** model element and its particular attributes and/or references. The purpose of this clause is to: provide a library of the **SCE** element depictions, and to provide an unambiguous resolution between the referenced **SCE** model element [*~~SCEElement~~Element*] and their depiction. Depiction resolution tables are provided below for both *~~SCEShape~~Shape* and *~~SCEEdge~~Edge*.

11.4.1 Labels

Both *~~SCEShape~~Shape* and *~~SCEEdge~~Edge* may have labels (its name attribute) placed on the shape/edge, or above or below the shape/edge, in any direction or location, depending on the preference of the modeler or modeling tool vendor.

Labels are optional for *~~SCEShape~~Shape* and *~~SCEEdge~~Edge*. When there is a label, the position of the label is specified by the bounds of the *~~SCELabel~~Label* of the *~~SCEShape~~Shape* or *~~SCEEdge~~Edge*. Simply put, label visibility is defined by the presence of the *~~SCELabel~~Label* element.

The bounds of the *~~SCELabel~~Label* are optional and always relative to the containing *~~SCEDiagram~~Diagram's~~SCEDiagram's~~* origin point. The depiction resolution tables provided below exemplify default label positions if no bounds are provided for the *~~SCELabel~~Label* (for *~~SCEShape~~Shape* kinds and *~~SCEEdge~~Edge* kinds (see sections above)).

When the *~~SCELabel~~Label* is contained in a *~~SCEShape~~Shape*, the text to display is the name of the *~~SCEElement~~BaseElement*.

11.4.2 ~~SCEShape~~Shape Resolution

~~SCEShape~~Shape can be used to represent a **Text Annotation** or a **Group**.

Commented [SW260]: This text was updated for the resolution of Issue SCE-115/SCE-116. Fix typo in labelHorizontalAlignment property name.

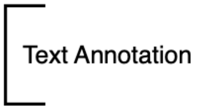
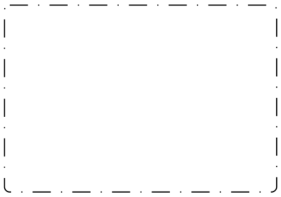
11.4.2.1 Diagram Artifacts

The **Association** element is included in the **SCE** metamodel as a *DiagramArtifact*. However, its notation is rendered through an ~~*SCEEdge*~~*Edge* (see section below).

Commented [SAW262]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

The following table presents the depiction resolutions for *DiagramArtifacts*:

~~Table 32.~~Table 28. Depiction Resolution of DiagramArtifacts

SCE Element	Depiction
TextAnnotation	
Group	

11.4.3 ~~SCEEdge~~Edge Resolution

~~*SCEEdge*~~*Edge* can be used to represent an **Association**.

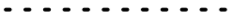
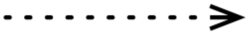
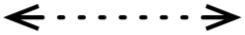
11.4.3.1 Association

Although an **Assocation** is placed in the **SCE** metamodel as a *DiagramArtifact*, its notation will be rendered with a ~~*SCEEdge*~~*Edge*. When the ~~*SCEEdge*~~*Edge* depicts an **Association**, its ~~*SCEElement*~~*BaseElement* MUST be specified.

Commented [SAW263]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

The following table presents the depiction resolutions for an **Association**:

~~Table 33.~~Table 29. Depiction Resolution of Association

SCE Element	Depiction
Association where associationDirection is none.	
Association where associationDirection is one.	
Association where associationDirection is both.	

Annex A: Mapping to BPMN

The elements of **SCE** are not current available for use by **BPMN**. At some point, the **BPMN** specifications may be updated to enable their utilization of **SCE** elements. As mentioned above, the design and structure of **SCE** is based on the design and structure of BPM+ specifications like **BPMN**. However, there are some differences and additions to **SCE** when compared to the **BPMN**. If there is not an exact match between an element in **BPMN** and a corresponding element in **SCE**, then a mapping will be defined.

~~Table 34.~~Table 30. Mapping to/from BPMN Base Element/Root Element

BPMN Element/Property	SCE Element/Property
BaseElement	~~*SCEElement*~~*BaseElement*
BaseElement.id	~~*SCEElement*~~*BaseElement*.id~~.identifier~~
Not used in **BPMN** BaseElement. The name property is included in specific BPMN elements that may have a name.	~~*SCEElement*~~*BaseElement*.name
Not included in **BPMN**.	~~*SCEElement*~~*BaseElement*.aliasIds~~.DaliasID~~
~~Not included in **BPMN**.~~	~~SCEElement.humanID~~
RootElement (extends BaseElement with no additional properties)	*RootElement*~~Not in **SCE**. *SCEElement* would be a substitute.~~

Commented [SW264]: This text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues.

Commented [SW265]: This text was edited for the resolution of Issue SCE-48/SCE-82. removing humanId and updating aliasId

Commented [SW266]: This table row was deleted for the resolution of Issue SCE-48/SCE-82. removing humanId and updating aliasId

Commented [SW267]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

~~Table 35.~~Table 31. Mapping to/from BPMN Definitions

BPMN Element/Property	SCE Element/Property
Definitions	~~*SCEModel*~~*Model*~~*Definitions*SCEDefinitions~~
Definitions.name	See ~~*SCEElement*~~*BaseElement*.name
Definitions.targetNamespace	~~*SCEModel*~~*Model*~~*SCEDefinitions*~~.targetNamespace
Definitions.expressionLanguage	~~*SCEModel*~~*Model*.expressionLanguage ~~Not in **SCE** since expressions are not included. This is **BPMN** specific metadata.~~
Definitions.typeLanguage	~~*SCEModel*~~*Model*.typeLanguage ~~Not in **SCE** since expressions are not included. This is **BPMN** specific metadata.~~
Definitions.exporter	~~*SCEPackage*~~*Model*~~SCEDefinitions~~.exporter
Definitions.exporterVersion	~~*SCEPackage*~~*Model*~~*SCEDefinitions*~~.exporterVersion
Not included in **BPMN**	~~*SCEPackage*~~*Package*.tags~~SCEDefinitions.tag~~
Not included in **BPMN**	~~*SCEPackage*~~*Package*~~SCEDefinitions~~.version
Not included in **BPMN**	~~*SCEPackage*~~*Package*~~*SCEDefinitions*~~.versionDate

Commented [SW268]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW269]: This text was updated for the resolution of Issue SCE-105/SCE-106. changing tag to tags

Commented [SW270]: Changes to this table were for the resolution of Issue SCE-40/SCE-41. Making SCE backwards compatible to older BPM+ languages.

Annex B: Mapping to CMMN

The elements of **SCE** are not current available for use by **CMMN**. At some point, the **CMMN** specifications may be updated to enable their utilization of **SCE** elements. As mentioned above, the design and structure of **SCE** is based on the design and structure of BPM+ specifications like **CMMN**. However, there are some differences and additions to **SCE** when compared to the **CMMN**. If there is not an exact match between an element in **CMMN** and a corresponding element in **SCE**, then a mapping will be defined.

~~**Table 36.**~~**Table 32.** **Mapping to/from CMMN CMMNElement**

CMMN Element/Property	SCE Element/Property
CMMNElement	~~*SCEElement*~~*BaseElement*
CMMNElement.id	~~*SCEElement*~~*BaseElement*.id~~.identifier~~
Not used in CMMNElement. The name property is included in specific **CMMN** elements that may have a name.	~~*SCEElement*~~*BaseElement*.name
Not included in **CMMN**.	~~*SCEElement*~~*BaseElement*.aliasIds~~.DaliasID~~
RootElement (extends BaseElement with no additional properties)~~Not included in **CMMN**.~~	*RootElement*~~SCEElement.humanID~~

Commented [SW271]: This text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues.

Commented [SW272]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW273]: This text was edited for the resolution of Issue SCE-48/SCE-82. removing humanId and updating aliasId

Commented [SW274]: This table row was deleted for the resolution of Issue SCE-48/SCE-82. removing humanId and updating aliasId

~~**Table 37.**~~**Table 33.** **Mapping to/from CMMN Definitions**

CMMN Element/Property	SCE Element/Property
Definitions	~~*SCEModel*~~*Model*~~SCEDefinitions~~
Definitions.name	See ~~SCEElement~~Element.name
Definitions.targetNamespace	~~*SCEModel*~~*Model*~~*SCEDefinitions*~~.targetNamespace
Definitions.expressionLanguage	~~*SCEModel*~~*Model*.expressionLanguage ~~Not in **SCE**. This is **CMMN** specific metadata.~~
Definitions.typeLanguage	~~*SCEModel*~~*Model*.typeLanguage
Definitions.exporter	~~*SCEPackage*~~*Model*~~SCEDefinitions~~.exporter
Definitions.exporterVersion	~~*SCEPackage*~~*Model*~~*SCEDefinitions*~~.exporterVersion
Definitions.author	Not in **SCE**. This is **CMMN** specific metadata~~, but could be provided by **PPMN**~~.
Definitions.creationDate	Not in **SCE**. This is **CMMN** specific metadata~~, but could be provided by **PPMN**~~.
Not included in **CMMN**	~~*SCEPackage*~~*Package*.tags~~SCEDefinitions.tag~~
Not included in **CMMN**	~~*SCEPackage*~~*Package*~~SCEDefinitions~~.version
Not included in **CMMN**	~~*SCEPackage*~~*Package*~~*SCEDefinitions*~~.versionDate

Commented [SW275]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

Commented [SW276]: This text updated for the resolution of Issue SCE-119/SCE-120. Remove references to Standards not yet completed.

Commented [SW277]: This text was updated for the resolution of Issue SCE-105/SCE-106. changing tag to tags

Commented [SW278]: Changes to this table were for the resolution of Issue SCE-40/SCE-41. Making SCE backwards compatible to older BPM+ languages.

Commented [SAW279]: The following text was updated for the resolution of Issue SCE-90/SCE-95. The removal of the "SCE" prefix for model elements.

Annex C: Mapping to DMN

The elements of **SCE** are not current available for use by **DMN**. At some point, the **DMN** specification may be updated to enable their utilization of **SCE** elements. As mentioned above, the design and structure of **SCE** is based on the design and structure of BPM+ specifications like **DMN**. However, there are some differences and additions to **SCE** when compared to the **DMN**. If there is not an exact match between an element in **DMN** and a corresponding element in **SCE**, then a mapping will be defined.

~~Table 38.~~Table 34. **Mapping to/from DMN DMNElement/NamedElement**

DMN Element/Property	SCE Element/Property
DMNElement	~~*SCEElement*~~*BaseElement*
DMNElement.id	~~*SCEElement*~~*BaseElement*.id~~.identifier~~
DMNElement.Description	**SCE** Documentation.body
DMNElement.Label	**SCE** Category.name
Not used in **DMN** DMNElement. The name property is included in specific BPMN elements that may have a name.	~~*SCEElement*~~*BaseElement*.name
Not included in **DMN**.	~~*SCEElement*~~*BaseElement*.aliasIds~~.DaliasID~~
~~Not included in **DMN**.~~	~~SCEElement.humanID~~
NamedElement (extends DMNElement)	Not in **SCE**. ~~*SCEElement*~~*BaseElement* would be a substitute.
NamedElement.name	~~*SCEElement*~~*BaseElement*.name

~~Table 39.~~Table 35. **Mapping to/from DMN Definitions**

DMN Element/Property	SCE Element/Property
Definitions	~~*SCEModel*~~*Model*~~SCEDefinitions~~
Definitions.namespace	~~*SCEModel*~~*Model*~~*SCEDefinitions*~~.targetNamespace
Definitions.expressionLanguage	~~*SCEModel*~~*Model*.expressionLanguage ~~Not in **SCE**. This is **DMN** specific metadata.~~
Definitions.typeLanguage	~~*SCEModel*~~*Model*.typeLanguage ~~Not in **SCE**. This is **DMN** specific metadata.~~
Definitions.exporter	~~*SCEPackage*~~*Model*~~SCEDefinitions~~.exporter
Definitions.exporterVersion	~~*SCEPackage*~~*Model*~~*SCEDefinitions*~~.exporterVersion
Not included in **DMN**	~~*SCEPackage*~~*Package*.tags~~SCEDefinitions.tag~~
Not included in **DMN**	~~*SCEPackage*~~*Package*~~SCEDefinitions~~.version
Not included in **DMN**	~~*SCEPackage*~~*Package*~~*SCEDefinitions*~~.versionDate

Commented [SW280]: This text updated for the resolution of Issue SCE-27/SCE-30. Editorial issues.

Commented [SW281]: This text was edited for the resolution of Issue SCE-48/SCE-82. removing humanId and updating aliasId

Commented [SW282]: This table row was deleted for the resolution of Issue SCE-48/SCE-82. removing humanId and updating aliasId

Commented [SW283]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW284]: This text was updated for the resolution of Issue SCE-96/SCE-97. More work for backwards compatibility.

Commented [SW285]: This text was updated for the resolution of Issue SCE-105/SCE-106. changing tag to tags

Commented [SW286]: Changes to this table were for the resolution of Issue SCE-40/SCE-41. Making SCE backwards compatible to older BPM+ languages.