This OMG document replaces the submission document (bmi/21-12-09). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome and should be directed to issues@omg.org by September 23, 2022.

You may view the pending issues for this specification from the OMG revision issues web page https://issues.omg.org/issues/lists.

The FTF Recommendation and Report for this specification will be published in December 2022. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.
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

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

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

CORBA®, CORBA logos®, FIBO®, Financial Industry Business Ontology®, FINANCIAL INSTRUMENT GLOBAL IDENTIFIER®, IIOP®, IMM®, Model Driven Architecture®, MDA®, Object Management Group®, OMG®, OMG Logo®, SoaML®, SOAML®, SysML®, UAF®, Unified Modeling Language®, UML®, UML Cube Logo®, VSIPL®, and XMI® are registered trademarks of the Object Management Group, Inc.

For a complete list of trademarks, see: http://www.omg.org/legal/tm_list.htm. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.
Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

**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 https://www.omg.org, under Documents, Report a Bug/Issue.
# Table of Contents

1. **Scope** ..................................................................................................................1
2. **Conformance** .......................................................................................................1
3. **References** .........................................................................................................1
   3.1 Normative References .........................................................................................1
   3.2 Non-normative References .................................................................................1
4. **Terms and Definitions** ........................................................................................2
5. **Symbols** .............................................................................................................2
6. **Additional Information** .......................................................................................2
   6.1 Conventions .......................................................................................................2
   6.2 Typographical and Linguistic Conventions and Style .........................................2
   6.3 Display of Metamodel Diagrams ........................................................................3
   6.4 Use of Text, Color, Size, and Lines in a Diagram .............................................4
   6.5 Abbreviations ....................................................................................................4
   6.6 Structure of this Document ...............................................................................5
   6.7 Acknowledgements ............................................................................................5
7. **Overview** ............................................................................................................5
8. **SCE Metamodel** ..................................................................................................6
   8.1 SCE Core Elements ............................................................................................6
     8.1.1 SCERootElement ...........................................................................................7
     8.1.2 SCEElement ..................................................................................................8
     8.1.3 ElementType ..................................................................................................9
     8.1.4 TypedElement ...............................................................................................9
     8.1.5 Packaging .....................................................................................................10
       8.1.5.1 SCEPackage ............................................................................................11
       8.1.5.2 SCEModelPackage ................................................................................13
       8.1.5.3 SCEModel ...............................................................................................15
       8.1.5.4 SCEDefinitions ......................................................................................16
       8.1.5.5 SCEInstances ...........................................................................................18
       8.1.5.6 SCEProfile ...............................................................................................19
   8.2 Annotations ..........................................................................................................20
     8.2.1 Annotation ....................................................................................................20
     8.2.2 Attachment ..................................................................................................21
     8.2.3 Category .......................................................................................................21
     8.2.4 Documentation ............................................................................................23
   8.3 External Relationships .........................................................................................24
     8.3.1 ExternalRelationship ....................................................................................24
     8.3.2 RelationshipDirection ................................................................................25
     8.3.3 Import ..........................................................................................................25
   8.4 Internal Relationships .........................................................................................26
     8.4.1 ElementRelationship ....................................................................................27
     8.4.2 ElementRelationshipType ............................................................................28
     8.4.3 RelationshipKind .........................................................................................29
   8.5 BPM+ Modeling ..................................................................................................31
     8.5.1 ModelArtifact ..............................................................................................31
# Table of Figures

- **Figure 1:** SCE Packages .................................................................6
- **Figure 2:** SCE High-Level Elements ..............................................7
- **Figure 3:** The SCEElement Metamodel ...........................................8
- **Figure 4:** The SCE Packaging Elements Metamodel .....................10
- **Figure 5:** The SCE Packaging Elements Metamodel (Details) ........11
- **Figure 6:** The SCEPackage Metamodel .........................................12
- **Figure 7:** The SCEModelPackage Metamodel ...............................14
- **Figure 8:** The SCEModel Metamodel ............................................15
- **Figure 9:** The SCEDefinitions Metamodel .....................................17
- **Figure 10:** The SCEInstances Metamodel .......................................18
- **Figure 11:** Annotations ..................................................................20
- **Figure 12:** An Example of a Groups referencing Categories (in an UML Object Diagram) .22
- **Figure 13:** An Example of a Parent and Children Categories (in an UML Object Diagram) .22
- **Figure 14:** The External Relationships Metamodel .........................24
- **Figure 15:** The Internal Relationships Metamodel .........................27
- **Figure 16:** The RelationshipKind Metamodel .................................30
- **Figure 17:** The ModelArtifact Metamodel ......................................32
- **Figure 18:** An Association ..............................................................33
- **Figure 19:** An Association Used with a Text Annotation ...............33
- **Figure 20:** A Group .......................................................................35
- **Figure 21:** A Text Annotation .........................................................36
- **Figure 22:** The SCEVocabulary Metamodel .................................37
- **Figure 23:** An Example of a Semantic Reference within a SDMN Model .................................................................38
- **Figure 24:** The RelationshipKinds Instance Model .......................40
- **Figure 25:** The SCEDI Metamodel ................................................44
- **Figure 26:** The SCEDiagram Metamodel ........................................45
- **Figure 27:** The SCEDiagramElement Metamodel .........................46
- **Figure 28:** The SCESShape Metamodel .........................................47
- **Figure 29:** The SCEEdge Metamodel ............................................48
- **Figure 30:** The SCELabel Metamodel ............................................49
- **Figure 31:** The SCEStyle Metamodel ............................................50
Table of Tables

Table 1. Submission Requirements ................................................................. Error! Bookmark not defined.
Table 2. Glossary ........................................................................................................... 2
Table 3. Acronyms ........................................................................................................ 4
Table 4. SCERootElement Attributes and/or Associations ........................................... 7
Table 5. SCEElement Attributes and/or Associations .................................................. 9
Table 6. TypedElement Attributes and/or Associations ............................................. 10
Table 7. SCEPackage Attributes and/or Associations ................................................ 12
Table 8. SCEModelPackage Attributes and/or Associations ..................................... 15
Table 9. SCEModel Attributes and/or Associations .................................................. 16
Table 10. SCEDefinitions Attributes and/or Associations ......................................... 17
Table 11. SCEInstances Attributes and/or Associations ............................................. 19
Table 12. Attachment Attributes and/or Associations ............................................... 21
Table 13. Category Attributes and/or Associations ................................................... 23
Table 14. Documentation Attributes and/or Associations ......................................... 23
Table 15. ExternalRelationship Attributes and/or Associations ................................ 25
Table 16. RelationshipDirection Literals .................................................................... 25
Table 17. Import Attributes and/or Associations ...................................................... 26
Table 18. ElementRelationship Attributes and/or Associations ................................ 28
Table 19. ElementRelationshipType Attributes and/or Associations ....................... 28
Table 20. RelationshipKind Instances ....................................................................... 31
Table 21. Association Attributes and/or Associations ............................................... 34
Table 22. AssociationDirection Literals ..................................................................... 34
Table 23. TextAnnotation Attributes and/or Associations ......................................... 36
Table 24. SemanticReference Attributes and/or Associations .................................... 39
Table 25. SCEVocabulary Attributes and/or Associations ........................................ 39
Table 26. RelationshipKind Instances ....................................................................... 41
Table 27. SCEDI Attributes and/or Associations .......................................................... 44
Table 28. SCEDiagram Attributes and/or Associations ............................................. 45
Table 29. SCEDiagramElement Attributes and/or Associations .................................. 47
Table 30. SCEEdge Attributes and/or Associations .................................................... 49
Table 31. SCELabel Attributes and/or Associations .................................................... 49
Table 32. SCEStyle Attributes and/or Associations .................................................... 51
Table 33. Depiction Resolution of DiagramArtifacts ................................................... 52
Table 34. Depiction Resolution of Association ............................................................. 53
Table 35. Mapping to/from BPMN Base Element/Root Element ................................. 54
Table 36. Mapping to/from BPMN Definitions ............................................................. 54
Table 37. Mapping to/from CMMN CMMNElement .................................................... 55
Table 38. Mapping to/from CMMN Definitions ............................................................ 55
Table 39. Mapping to/from DMN DMNElement/NamedElement .................................. 56
Table 40. Mapping to/from DMN Definitions ............................................................... 56
Annexes

Annex C: Mapping to BPMN .................................................................54
Annex D: Mapping to CMMN ..............................................................54
Annex E: Mapping to DMN .................................................................55
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 specifications, BKPMN, PPMN, and SDMN, are 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.

2 Conformance

SCE 1.0 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 Situational Data Model and Notation (SDMN), and the Pedigree and Provenance Model and Notation (PPMN) specifications are dependent on SCE 1.0.

Software that claims compliance or conformance to any specification that is dependent on SCE 1.0 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™)

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.

4 Terms and Definitions

The table below presents a glossary for this specification:

Table 1. Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
<td>A CMMN element that is a proceeding that involves actions taken regarding a subject in a particular situation to achieve a desired outcome.</td>
</tr>
<tr>
<td>DataItem</td>
<td>A SDMN DataItem represents a common definition and structure for the data handling elements of the other BPM+ models.</td>
</tr>
<tr>
<td>DataState</td>
<td>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.</td>
</tr>
<tr>
<td>Decision</td>
<td>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.</td>
</tr>
<tr>
<td>Process</td>
<td>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.</td>
</tr>
</tbody>
</table>

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,” “MAY,” “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:
  o <none> — exactly once
  o [0..1] — 0 or 1
  o [0..*] — 0 or more
  o [1..*] — 1 or more

• Attributes separated by | and grouped within { and } — alternative values
  o <value> — default value
  o <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 1: SCE Metamodel Color-Coding**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Example Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCE General Class</td>
<td>These elements elements include the owner of the language (SCE) in parentheses 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).</td>
<td><img src="image" alt="NamedElement (SCE.Core)" /></td>
</tr>
<tr>
<td>SCE General Class (focus of diagram)</td>
<td>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.</td>
<td><img src="image" alt="SpecificationPackage (SCE.Core)" /></td>
</tr>
<tr>
<td>External Class</td>
<td>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 parentheses below the element name and these elements are color-coded light-gray. (see figure to the right).</td>
<td><img src="image" alt="Shape (SCEDI.DI)" /></td>
</tr>
</tbody>
</table>
### 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 definition, that are used in this specification:

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BKPMN</td>
<td>BPM+ Knowledge Package Model and Notation</td>
</tr>
<tr>
<td>BPM+</td>
<td>Business Process Management Plus</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Model and Notation</td>
</tr>
<tr>
<td>CMMN</td>
<td>Case Management Model and Notation</td>
</tr>
<tr>
<td>DC</td>
<td>Diagram Commons</td>
</tr>
<tr>
<td>DD</td>
<td>Diagram Definition</td>
</tr>
<tr>
<td>DI</td>
<td>Diagram Interchange</td>
</tr>
<tr>
<td>DMN</td>
<td>Decision Model and Notation</td>
</tr>
<tr>
<td>MOF</td>
<td>Meta Object Facility</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>PPMN</td>
<td>Provenance and Pedigree Model and Notation</td>
</tr>
<tr>
<td>RFC</td>
<td>Remote Function Call</td>
</tr>
<tr>
<td>SCE</td>
<td>Specification Common Elements</td>
</tr>
<tr>
<td>SCEDI</td>
<td>Specification Common Elements Diagram Interchange</td>
</tr>
<tr>
<td>SDMN</td>
<td>Shared Data Model and Notation</td>
</tr>
</tbody>
</table>
6.6 Structure of this Document
This document provides a brief introduction to SCE and its purpose (see the section entitled “Error! Reference source not found.”). 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

Submitting Organizations (RFP Process)
- Auxilium Technology Group, LLC
- BPM Advantage Consulting, Inc.

Supporting Organizations (RFP Process)
The following organizations support this specification but are not formal submitters:
- Airbus Group
- BookZurman, Inc.
- Camunda Services GmbH
- Department of Veterans Affairs
- FICO
- Mayo Clinic
- MDIX, Inc.
- Red Hat
- Thematix Partners, LLC
- Trisotech
- XZYOS, LLC

Special Acknowledgements
The following persons were members of the core teams that contributed to the content of this specification: Claude Baudoin, John Butler, Keith Butler, Lloyd Duggen, Denis Gagne, Eder Ignatowicz, Peter Haug, Elisa Kendall, Matteo Mortari, Falko Menge, Sean Muir, Robert Lario, Ken Lord, Peter Rivett, Keith Salzman, Jane Shellum, Davide Sottara, and Stephen A. White.

7 Overview
The idea for defining a Specification Core Element Model (SCE) occurred during the development of the BKPMN and SDMN specifications. 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 shared a common set of 8 elements and their attributes. 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 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.

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 Vocabularies.

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

![SCE Packages](image)

8.1 SCE Core Elements

There are two core abstract elements that make up SCE with a few supporting elements. The core elements are: SCERootElement and SCEElement. There are six 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:
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 3. SCERootElement Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliasID : String [0..*]</td>
<td>Various alternative identifiers for this Element. Generally, these will be set by tools, but one of them (the humanId), in particular, may be set by the modeler.</td>
</tr>
</tbody>
</table>
**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.

**id** : String [1]

This 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 identify this requirement.

### 8.1.2 SCEElement

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

The following figure presents the metamodel for SCEElement:

**Figure 3:** The SCEElement Metamodel

**Generalizations**

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 SCEElement:

Table 4. SCEElement Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachment : Attachment [0..*]</td>
<td>This association is used to annotate any concrete specialization of SCEElement with descriptions and other documentation.</td>
</tr>
<tr>
<td>categoryRef : Category [0..*]</td>
<td>This association is used to categorize any concrete specialization of SCEElement. A Category has user-defined semantics, which can be used for documentation or analysis purposes.</td>
</tr>
<tr>
<td>documentation : Documentation [0..*]</td>
<td>This association is used to annotate any concrete specialization of SCEElement with descriptions and other documentation.</td>
</tr>
<tr>
<td>semanticReferenceRef : SemanticReference [0..*]</td>
<td>A concrete SCEElement can reference zero or more SemanticReference elements.</td>
</tr>
</tbody>
</table>

8.1.3 ElementType
A kind of SCEElement 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 (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 ElementType element does not have any additional attributes and/or associations.

8.1.4 TypedElement
A kind of SCEElement 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 (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 *TypedElement*:

**Table 5. TypedElement Attributes and/or Associations**

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

### 8.1.5 Packaging

*SCE* provides six elements 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 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](image)
The following figure presents the attributes and associations for the SCE packaging elements, including more details about the elements they contain:

![SCE Packaging Elements Metamodel](image)

**Figure 5: The SCE Packaging Elements Metamodel (Details)**

### 8.1.5.1 SCEPackage

*SCEPackage* 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-classes (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 *SCEModelPackage* (see below) is the top-level package used for distribution of the content of a modeling language.

Note: a **targetNamespace** 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 tSCEPackage type for the SCE XSD.
The following figure presents the metamodel for \textit{SCEPackage}: 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{scepackage_metamodel.png}
\caption{The SCEPackage Metamodel}
\end{figure}

\textbf{Generalizations}

The \textit{SCEPackage} element inherits the attributes and/or associations of:

- \textit{SCEEelement} (see the section entitled “SCEEelement” for more information).

Further, the \textit{SCEEelement} element inherits the attributes and/or associations of:

- \textit{SCERootElement} (see the section entitled “SCERootElement” for more information).

\textbf{Properties}

The following table presents the additional attributes and/or associations for \textit{SCEPackage}:

\begin{table}[h]
\centering
\caption{SCEPackage Attributes and/or Associations}
\begin{tabular}{|c|c|}
\hline
\textbf{Property/Association} & \textbf{Description} \\
\hline
\texttt{containedPackage} : SCEPackage [0..*] & This is a list of all the sub-packages \textit{SCEPackage}. This provides the capability for all specializations of \textit{SCEPackage} to include sub-packages. This is a subset of the \texttt{element} association of the \textit{SCEPackage} element. \\
\hline
\end{tabular}
\end{table}
<table>
<thead>
<tr>
<th>element</th>
<th>SCERootElement [0..*]</th>
<th>This is a list of all the SCERootElements contained within a SCEModelPackage. Many elements will be identified through additional associations that subset this property (see figure above).</th>
</tr>
</thead>
<tbody>
<tr>
<td>exporter</td>
<td>String [0..1]</td>
<td>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.</td>
</tr>
<tr>
<td>exporterVersion</td>
<td>String [0..1]</td>
<td>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.</td>
</tr>
<tr>
<td>import</td>
<td>Import [0..*]</td>
<td>This attribute is used to import externally defined elements and make them available for use by elements within a concrete specialization of an SCEPackage.</td>
</tr>
<tr>
<td>tag</td>
<td>String [0..*]</td>
<td>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 SCEModelPackage, for example.</td>
</tr>
<tr>
<td>version</td>
<td>String [0..1]</td>
<td>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.</td>
</tr>
<tr>
<td>versionDate</td>
<td>date [0..1]</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 8.1.5.2 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 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 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 SCEModelPackage 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 concrete modeling package.
The following figure presents the metamodel for \textit{SCEModelPackage}:

![SCEModelPackage Metamodel Diagram]

**Figure 7: The SCEModelPackage Metamodel**

**Generalizations**

The \textit{SCEModelPackage} element inherits the attributes and/or associations of:

- \textit{SCEPackage} (see the section entitled “\textit{SCEPackage}” for more information).

Further, the \textit{SCEPackage} element inherits the attributes and/or associations of:

- \textit{SCEElement} (see the section entitled “\textit{SCEElement}” for more information).

Further, the \textit{SCEElement} element inherits the attributes and/or associations of:

- \textit{SCERootElement} (see the section entitled “\textit{SCERootElement}” for more information).

**Properties**

The following table presents the additional attributes and/or associations for \textit{SCEModelPackage}:
Table 7. SCEModelPackage Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>model : SCEModel [1]</td>
<td>This the SCEModel sub-package contained within a SCEModelPackage. This is a subset of the containedPackage association of the SCEPackage element.</td>
</tr>
<tr>
<td>presentation : SCEDI [0..1]</td>
<td>This attribute contains the Diagram Interchange information contained within this SCEModelPackage.</td>
</tr>
</tbody>
</table>

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 concrete modeling package.

The following figure presents the metamodel for SCEModel:

![The SCEModel Metamodel](image)

Figure 8: The SCEModel Metamodel

Generalizations

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).

Properties

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

Table 8. SCEModel Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>category : Category [0..*]</td>
<td>This is a list of all the Categories contained within a concrete specialization of SCEModel.</td>
</tr>
<tr>
<td>definitions: SCEDefinitions [0..*]</td>
<td>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.</td>
</tr>
<tr>
<td>externalRelationship : ExternalRelationship [0..*]</td>
<td>This is a list of all the ExternalRelationships contained within a concrete specialization of SCEDefinitions.</td>
</tr>
<tr>
<td>instances : SCEInstances [0..*]</td>
<td>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.</td>
</tr>
<tr>
<td>profile : SCEProfile [0..*]</td>
<td>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.</td>
</tr>
<tr>
<td>sceVocabulary : SCEVocabulary [0..*]</td>
<td>This is a list of terms (SemanticReferences) that can be used to define the elements of a concrete specialization of SCEModel.</td>
</tr>
</tbody>
</table>

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 concrete modeling package.

The following figure presents the metamodel for SCEDefinitions:
Generalizations

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).

Properties

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

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containedDefinitions : SCEDefinitions [0..*]</td>
<td>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.</td>
</tr>
</tbody>
</table>
### 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 concrete modeling package.

The following figure presents the metamodel for `SCEInstances`:

![The SCEInstances Metamodel](image)

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>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>containedInstances : SCEInstances [0..*]</td>
<td>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.</td>
</tr>
<tr>
<td>definitionsRef : SCEDefinitions [0..*]</td>
<td>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.</td>
</tr>
<tr>
<td>elementRelationship : ElementRelationship [0..*]</td>
<td>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.</td>
</tr>
<tr>
<td>modelArtifact : ModelArtifact [0..*]</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 concrete 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.

### 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*-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 concrete types of *Annotations*: *Attachments*, *Categories*, and *Documentation*.

The following figure shows the metamodel for *Annotations*.

![Annotations Metamodel](image)

**Figure 11: 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
The Annotation element does not have any additional attributes and/or associations.

8.2.2 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. Thus, any concrete element within a model that is dependent on SCE MAY have one or more Attachments.

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:
- 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 Attachment:

Table 11. Attachment Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachmentLocation : URI [1]</td>
<td>This attribute identifies the URI location of the attachment.</td>
</tr>
</tbody>
</table>

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.
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 Responsibilities” through that Category’s association with the “Lifestyle Modification” Category, which is the parent of the Category directly associated with the Recommendation.

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

Table 12. Category Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>child : Category [0..*]</td>
<td>This association allows the nesting of Categories. A Category MAY have more than one child Category.</td>
</tr>
<tr>
<td>parentRef : Category [0..1]</td>
<td>This association allows the nesting of Categories. A Category MAY be a parent for more than one Category.</td>
</tr>
</tbody>
</table>

8.2.4 Documentation
The Documentation element provides a place for model developers to provide descriptive information about an model element.

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

Generalizations
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 “SCERootElement” for more information).

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

Table 13. Documentation Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>body : String [1]</td>
<td>This attribute is used to capture the text descriptions of any concrete element within a model that is dependent on SCE.</td>
</tr>
<tr>
<td>language : String [1]</td>
<td>The named language can be a natural language, in which case the body is an informal representation, or an artificial language, in which case the body is expected to be a formal, machine-parsable representation.</td>
</tr>
</tbody>
</table>
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 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 ExternalRelationship metamodel diagram.

![ExternalRelationship Metamodel Diagram](image)

Figure 14: The External Relationships Metamodel

8.3.1 ExternalRelationship

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

Generalizations
The `ExternalRelationship` 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 `ExternalRelationship`:

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

### 8.3.2 RelationshipDirection

This enumeration list specifies the direction of the relationship.

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

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backward</td>
<td>This literal specifies that the <code>ExternalRelationship</code> is in the direction from the target to the source.</td>
</tr>
<tr>
<td>both</td>
<td>This literal specifies that the <code>ExternalRelationship</code> is in the direction from the target to the source and from the source to the target.</td>
</tr>
<tr>
<td>forward</td>
<td>This literal specifies that the <code>ExternalRelationship</code> is in the direction from the source to the target.</td>
</tr>
<tr>
<td>none</td>
<td>This literal specifies that the <code>ExternalRelationship</code> is in the direction from the target to the source.</td>
</tr>
</tbody>
</table>

### 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`.

**Generalizations**

The `Import` element inherits the attributes and/or associations of:
Properties

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

Table 16. Import Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>importType: URI [1]</td>
<td>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 <a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a> when importing XML Schema 1.0 documents, to <a href="http://www.w3.org/TR/wsdl20/">http://www.w3.org/TR/wsdl20/</a> when importing WSDL 2.0 documents, and <a href="http://www.omg.org/spec/BPMN/20100524/MODEL">http://www.omg.org/spec/BPMN/20100524/MODEL</a> 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. 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 <a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a> when importing XML Schema 1.0 documents, to <a href="http://www.w3.org/TR/wsdl20/">http://www.w3.org/TR/wsdl20/</a> when importing WSDL 2.0 documents, and <a href="http://www.omg.org/spec/BPMN/20100524/MODEL">http://www.omg.org/spec/BPMN/20100524/MODEL</a> 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.</td>
</tr>
<tr>
<td>location: URI [0..1]</td>
<td>Identifies the location of the imported element within the document identified by the importType.</td>
</tr>
<tr>
<td>namespace: URI [1]</td>
<td>Identifies the namespace of the imported element.</td>
</tr>
</tbody>
</table>

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 SDMNVocabulary for RelationshipKind):
8.4.1 ElementRelationship

A kind of relationships between two SCEElements. The RelationshipKind element identify specific types of relationships.

Generalizations

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

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

Further, the TypedElement 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 ElementRelationship:
Table 17. ElementRelationship Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceRef : SCEElement [1]</td>
<td>The source SCEElement of the relationship. If there is an ElementRelationshipType identified through the typeRef association, then the source must be a TypedElement.</td>
</tr>
<tr>
<td>targetRef : SCEElement [1]</td>
<td>The target concrete specialization of SCEElement of the relationship. If there is an ElementRelationshipType identified through the typeRef association, then the target must be a TypedElement.</td>
</tr>
<tr>
<td>relationshipKindRef : RelationshipKind [1]</td>
<td>A description of the type of the relationship. See RelationshipKind, below, for more details.</td>
</tr>
<tr>
<td>typeRef : ElementRelationshipType [0..1]</td>
<td>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.</td>
</tr>
</tbody>
</table>

8.4.2 ElementRelationshipType

A kind of ElementRelationship that specifies two ElementTypes (rather than 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 “ElementType” for more information).

Further, the ElementType 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 ElementRelationshipType:

Table 18. ElementRelationshipType Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceMultiplicity : String [0..1]</td>
<td>This attribute defines the minimum number of source SCEElements that may be the source for the ElementRelationship that identifies this ElementRelationshipType through its typeRef association.</td>
</tr>
<tr>
<td>sourceRef : ElementType [1]</td>
<td>The source ElementType of the relationship.</td>
</tr>
</tbody>
</table>
targetMultiplicity : String [0..1]  
This attribute defines the minimum number of target 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.

8.4.3 RelationshipKind
This class is a type of SemanticReference that serves as the terms for an SCEVocabulary that is 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).

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 the Library.

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

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

- `SemanticReference` (see the section entitled “SemanticReference” for more information).

Further, 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 RelationshipKind element does not have any additional attributes and/or associations.

Standard Terms Vocabulary

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

<table>
<thead>
<tr>
<th>Instance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composition</td>
<td>Composition indicates that the source element is composed of, in part, the target element. Other elements could be included in this composition.</td>
</tr>
<tr>
<td>Containment</td>
<td>Containment indicates that the source element is a container for the target element.</td>
</tr>
<tr>
<td>Correlation</td>
<td>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.</td>
</tr>
<tr>
<td>Dependency</td>
<td>Dependency indicates that target element is dependent in some way on the source element.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>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.</td>
</tr>
<tr>
<td>Reference</td>
<td>Reference indicates that source element references the target element.</td>
</tr>
<tr>
<td>Generalization</td>
<td>Generalization indicates that the source element is a generalization of the target element (which is based on and extends the source).</td>
</tr>
</tbody>
</table>

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. 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.

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: The ModelArtifact Metamodel](image)

**Generalizations**

The ModelArtifact 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 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 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.

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: An Association

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

---

The Order Set Data is maintained in a separate SDMN Model

---

Order Set Data Items

---

Figure 19: An Association Used with a Text Annotation

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” for more information).

Further, the ModelArtifact 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 Association:

Table 20. Association Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>associationDirection : AssociationDirection [1]</td>
<td>AssociationDirection is an attribute that defines whether or not the 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 line.</td>
</tr>
<tr>
<td>sourceRef : SCEElement [1]</td>
<td>The SCEElement that the Association is connecting from.</td>
</tr>
<tr>
<td>targetRef : SCEElement [1]</td>
<td>The SCEElement that the Association is connecting to.</td>
</tr>
</tbody>
</table>

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 21. AssociationDirection Literals

<table>
<thead>
<tr>
<th>Literal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>both</td>
<td>A value of “both” means that there SHALL be an arrowhead at both ends of the Association.</td>
</tr>
<tr>
<td>none</td>
<td>The default is “none” (no arrowhead).</td>
</tr>
<tr>
<td>one</td>
<td>A value of “one” means that the arrowhead SHALL be at the targetRef Object.</td>
</tr>
</tbody>
</table>

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.
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:

- *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 **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**

- **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: A Text Annotation

Generalizations

The TextAnnotation element inherits the attributes and/or associations of:
- DiagramArtifact (see the section entitled “DiagramArtifact” for more information).

Further, the DiagramArtifact 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 TextAnnotation:

Table 22. TextAnnotation Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>annotatedElementRef : SCEElement [0..*]</td>
<td>If the TextAnnotation is associated with (is the sourceRef of an Association) another model element, this association will identify the targetRef of the Association. It is derived from the connected Association element.</td>
</tr>
<tr>
<td>commentRef : Documentation [0..1]</td>
<td>CommentRef is one of two attributes that provides text that the modeler wishes to communicate to the reader of the model. The text within a commentRef references a Documentation element that is contained in SCEDefinitions. Thus, a particular commentRef may appear on multiple models. This association will also allow a TextAnnotation to display the Documentation of the diagram element that the TextAnnotation is associated with (is connected to by an Association). This attribute is optional, but if it is used, then the note attribute SHALL NOT be used.</td>
</tr>
<tr>
<td>language : String [0..1]</td>
<td>The named language can be a natural language, in which case the body is an informal representation, or an artificial language, in which case the body is expected to be a formal, machine-parsable representation. If the note attribute is used, then the language attribute is required.</td>
</tr>
</tbody>
</table>
**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**

Vocabularies (lists of terms) can be added to a model package of a modeling language dependent on SCE. SCEVocabularies are sets of terms defined by an external ontology. The terms link to formal definitions for the model elements that are created by the modeling language. The SemanticReference element is used to name the term provide a link to the definitions. SCEVocabularies are contained within an SCEModel package.

The following figure presents the metamodel for SCEVocabulary:

![SCEVocabulary Metamodel](image)

**Figure 22: The SCEVocabulary Metamodel**

**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.

![SnoMed](image)

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 23. SemanticReference Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conceptNamespace : URI [0..1]</td>
<td>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.</td>
</tr>
<tr>
<td>conceptURI : URI [0..1]</td>
<td>This attribute defines the URI location of the target of the SemanticReference.</td>
</tr>
</tbody>
</table>

8.6.2 SCEVocabulary

An SCEVocabulary is a list of terms, through the SemanticReference 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 SCEVocabularies can be defined. They are contained in an SCEModel.

Further, 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 SCEVocabularies into the appropriate enumerated lists. Since the SemanticReference element has a name and the links to external definitions are optional, the list (the “enumeration” SCEVocabulary) can be created before the specific external definitions are established.

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

Generalizations

The SCEVocabulary 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 SCEVocabulary:

Table 24. SCEVocabulary Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>term : SemanticReference [0..*]</td>
<td>The list of terms is a set of SemanticReferences to an external ontology.</td>
</tr>
</tbody>
</table>
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 an SCEVocabulary: RelationshipKinds which is provided by the SCE Library. The purpose of this vocabulary is to provide a set of standard terms, which are instances of the RelationshipKind element.

The RelationshipKind element is used to specify 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 element:

![The RelationshipKinds Instance Model](image)

Figure 24: The RelationshipKinds Instance Model

The following table presents a description for the included instances for RelationshipKind:
<table>
<thead>
<tr>
<th>Instance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composition</td>
<td>Composition indicates that the source element is composed of, in part, the target element. Other elements could be included in this composition.</td>
</tr>
<tr>
<td>Containment</td>
<td>Containment indicates that the source element is a container for the target element.</td>
</tr>
<tr>
<td>Correlation</td>
<td>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.</td>
</tr>
<tr>
<td>Dependency</td>
<td>Dependency indicates that target element is dependent in some way on the source element.</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>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.</td>
</tr>
<tr>
<td>Reference</td>
<td>Reference indicates that source element references the target element.</td>
</tr>
<tr>
<td>Generalization</td>
<td>Generalization indicates that the source element is a generalization of the target element (which is based on and extends the source).</td>
</tr>
</tbody>
</table>

### 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.

#### 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.

XML 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 `<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:Import>` element.

Each file SHALL declare a “namespace” that MAY differ between multiple files of one model.

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

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 support referencing by ID, across files, by utilizing an href attribute whose value must be a valid URI reference [RFC 3986] where the path components may be absolute or relative, the reference has no query component, and the fragment consists of the value of the id of the referenced SCE element.

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.).

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 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.

## 11.3 SCE Diagram Interchange Meta-Model

### 11.3.1 How to read this chapter

Clause 10.4 describes in detail the meta-model used to keep the layout and the look of SCE-dependent Diagrams. Clause 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.

### 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 [SCEDiagram] for interchange requires the specification of a collection of shapes [SCEShape] and edges [SCEEdge].

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 [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 is an ordered collection of mixed SCEShape(s) and SCEEdge(s). The order of the SCEShape(s) and SCEEdge(s) inside a SCEDiagram determines their Z-order (i.e., what is in front of what). SCEShape(s) and SCEEdge(s) that are meant to be depicted “on top” of other SCEShape(s) and SCEEdge(s) MUST appear after them in the SCEDiagram. Thus, the exporting tool MUST order all SCEShape(s) and SCEEdge(s) such that the desired depiction can be rendered.

### 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

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

The following figure shows the SCEDI metamodel diagram.
The SCEDI element does not inherit any attributes or associations from another element.

Properties

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

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagram : SCEDiagram [0..*]</td>
<td>A list of SCEDiagrams.</td>
</tr>
<tr>
<td>style : SCEStyle [0..*]</td>
<td>A list of shared SCEStyle that can be referenced by all SCE-dependent diagrams and SCEDiagramElement.</td>
</tr>
</tbody>
</table>

11.3.4.2 SCEDiagram

The abstract class SCEDiagram 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 define concrete diagrams based on SCEDiagram.

SCEDiagram is the container of SCEDiagramElement (SCEShape(s) and SCEEedge(s)). SCEDiagram cannot include other SCEDiagrams.

A SCEDiagram can define a SCEStyle 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 SCEDiagramElement contained in this SCEDiagram.

The SCEDiagram 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.

The following figure shows the SCEDiagram metamodel diagram.

---

Figure 25: The SCEDI Metamodel

Generalizations

The SCEDI element does not inherit any attributes or associations from another element.
Figure 26: The SCEDiagram Metamodel

Generalizations

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

- 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” for more information).

Properties

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

Table 27. SCEDiagram Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagramElement : SCEDiagramElement [0..*]</td>
<td>A list of SCEDiagramElements (SCEShape and SCEEdge) that are depicted in the SCE-dependent diagram.</td>
</tr>
<tr>
<td>diagramRef : SCEDiagram [1]</td>
<td>The diagram that the DI is representing.</td>
</tr>
</tbody>
</table>
### localStyle : SCEStyle [0..1]
- A `SCEStyle` that defines the default styling for this diagram. Properties defined in that style override the ones in the `sharedStyle`.

### sharedStyleRef : SCEStyle [0..*]
- A reference to a `SCEStyle` defined in the SCEDI that serves as the default styling of the `SCEDiagramElement` in the SCE-dependent diagram.

### size : DC:Dimension [0..1]
- The size of this diagram. If not specified, the SCE-dependent diagram is unbounded.

11.3.4.3 **SCEDiagramElement**

The `SCEDiagramElement` class is contained by the `SCDiagram` and is the base class for `SCEShape` and `SCEEdge`. `SCEDiagramElement` inherits its styling from its parent `SCDiagram`. In addition, it can refer to one of the shared `SCEStyle` defined in the SCEDI and/or it can define a local style. See section below for more details on styling.

`SCEDiagramElement` MAY also contain a `SCELabel` when it has a visible text label. If no `SCELabel` is defined, the `SCEDiagramElement` should be depicted without a label.

The following figure shows the `SCEDiagramElement` metamodel diagram.

![SCEDiagramElement Metamodel](image)

#### Generalizations

The `SCEDiagramElement` element inherits the attributes and/or associations of:

- `DiagramElement` (see the section entitled “DiagramElement” for more information).

#### Properties

The following table presents the additional attributes and/or associations for `SCEDiagramElement`:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>localStyle</td>
<td>A <code>SCEStyle</code> that defines the default styling for this diagram. Properties defined in that style override the ones in the <code>sharedStyle</code>.</td>
</tr>
<tr>
<td>sharedStyleRef</td>
<td>A reference to a <code>SCEStyle</code> defined in the SCEDI that serves as the default styling of the <code>SCEDiagramElement</code> in the SCE-dependent diagram.</td>
</tr>
<tr>
<td>size</td>
<td>The size of this diagram. If not specified, the SCE-dependent diagram is unbounded.</td>
</tr>
</tbody>
</table>
Table 28. SCEDiagramElement Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label : SCELabel [0..*]</td>
<td>An optional label when the SCE-dependent Element has a visible text label.</td>
</tr>
<tr>
<td>localStyle : SCEStyle [0..1]</td>
<td>A SCEStyle that defines the styling for this element.</td>
</tr>
<tr>
<td>sceElementRef : SCEElement [1]</td>
<td>A reference to the concrete instance of the SCEElement that is being depicted.</td>
</tr>
<tr>
<td>sharedStyleRef : SCEStyle [0..1]</td>
<td>A reference to a SCEStyle defined in the SCEDI.</td>
</tr>
</tbody>
</table>

11.3.4.4 SCEShape

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

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

SCEShape 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 metamodel diagram.

![Figure 28: The SCEShape Metamodel](image)

Generalizations

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

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

Further, the SCEDiagramElement element inherits the attributes and/or associations of:
• DiagramElement (see the section entitled “DiagramElement” for more information).

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

• Shape (see the section entitled “Shape” for more information).

Properties

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

11.3.4.5 SCEEdge

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

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

The following figure shows the SCEEdge metamodel diagram.

![SCEEdge Metamodel Diagram](image)

Figure 29: The SCEEdge Metamodel

Generalizations

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

• Edge (see the section entitled “Edge” for more information).

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

• SCEDiagramElement (see the section entitled “SCEDiagramElement” for more information).

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

• DiagramElement (see the section entitled “DiagramElement” for more information).

Properties

The following table presents the additional attributes and/or associations for SCEEdge:
Table 29. SCEEdge Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sourceElementRef : SCEDiagramElement [0..1]</td>
<td>The actual SCEDiagramElement this SCEEdge is connecting from. This MUST be specified when the SCEEdge has a source.</td>
</tr>
<tr>
<td>targetElementRef : SCEDiagramElement [0..1]</td>
<td>The actual SCEDiagramElement this SCEEdge is connecting to. This MUST be specified when the SCEEdge has a target.</td>
</tr>
</tbody>
</table>

11.3.4.6 SCELabel

SCELabel represents the depiction of some textual information about an element.

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

The following figure shows the SCELabel metamodel diagram.

Figure 30: The SCELabel Metamodel

Generalizations

The SCELabel element inherits the attributes and/or associations of:

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

Properties

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

Table 30. SCELabel Attributes and/or Associations

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text : String [0..1]</td>
<td>An optional pretty printed text that MUST be displayed instead of the SCEElement’s name if it is present.</td>
</tr>
</tbody>
</table>
### 11.3.4.7 SCEStyle

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

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

- The *SCEStyle* defined by the localStyle attribute of the *SCEDiagramElement*
- The *SCEStyle* referenced by the sharedStyle attribute of the *SCEDiagramElement*
- The *SCEStyle* defined by the localStyle attribute of the parent *SCEDiagram*
- The *SCEStyle* referenced by the sharedStyle attribute of the parent *SCEDiagram*

The default attribute value defined in *SCEStyle* attributes.

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

- *SCEDiagramElement* has a local *SCEStyle* that specifies the fillColor and strokeColor
- Its parent *SCEDiagram* defines a local *SCEStyle* that specifies the fillColor and fontColor

Then the resulting *SCEDiagramElement* should use:

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

---

**Figure 31:** The SCEStyle Metamodel
Generalizations
The SCEStyle 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:

<table>
<thead>
<tr>
<th>Property/Association</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fillColor : Color [0..1]</td>
<td>The color use to fill the shape. Doesn’t apply to SCEEdge. The default is white.</td>
</tr>
<tr>
<td>fontBold : boolean [0..1]</td>
<td>If the text should be displayed in Bold. The default is false.</td>
</tr>
<tr>
<td>fontColor : Color [0..1]</td>
<td>The color use to write the label. The default is black.</td>
</tr>
<tr>
<td>fontFamily : String [0..1]</td>
<td>A comma-separated list of Font Name that can be used to display the text. The default is Arial.</td>
</tr>
<tr>
<td>fontItalic : boolean [0..1]</td>
<td>If the text should be displayed in Italic. The default is false.</td>
</tr>
<tr>
<td>fontSize : Real [0..1]</td>
<td>The size in points of the font to use to display the text. The default is 8.</td>
</tr>
<tr>
<td>fontStrikeThrough : boolean[0..1]</td>
<td>If the text should be stroke through. The default is false.</td>
</tr>
<tr>
<td>fontUnderline : boolean [0..1]</td>
<td>If the text should be underlined. The default is false.</td>
</tr>
<tr>
<td>id : String [0..1]</td>
<td>A unique id for this style so it can be referenced. Only styles defined in the SCEDI can be referenced by SCEDiagramElement and SCEDiagram.</td>
</tr>
<tr>
<td>labelHorizontalAlignment : AlignmentKind [0..1]</td>
<td>How text should be positioned horizontally within the Label bounds. Default depends of the SCEDiagramElement the label is attached to (see section below).</td>
</tr>
<tr>
<td>labelVerticalAlignment : AlignmentKind [0..1]</td>
<td>How the text should be positioned vertically inside the Label bounds. Default depends of the SCEDiagramElement the label is attached to (see section below). Start means “top” and end means “bottom”.</td>
</tr>
<tr>
<td>strokeColor : Color [0..1]</td>
<td>The color use to draw the shape borders. The default is black.</td>
</tr>
</tbody>
</table>

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(s) and SCEEdge(s) in the SCEDiagram (see sections above). The SCEShape(s) and SCEEdge(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(s) and SCEEdge(s) MUST reference SCE model elements. If no SCEElement 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 or SCEEdge 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] and their depiction. Depiction resolution tables are provided below for both SCEShape and SCEEdge.

### 11.4.1 Labels

Both SCEShape and SCEEdge 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 and SCEEdge. When there is a label, the position of the label is specified by the bounds of the SCELabel of the SCEShape or SCEEdge. Simply put, label visibility is defined by the presence of the SCELabel element.

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

When the SCELabel is contained in a SCEShape, the text to display is the name of the SCEElement.

### 11.4.2 SCEShape Resolution

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

#### 11.4.2.1 Diagram Artifacts

The Association element is included in the SCE metamodel as a DiagramArtifact. However, its notation is rendered through a SCEEdge (see section below).

The following table presents the depiction resolutions for DiagramArtifacts:

<table>
<thead>
<tr>
<th>SCE Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextAnnotation</td>
<td><img src="image" alt="Text Annotation" /></td>
</tr>
<tr>
<td>Group</td>
<td><img src="image" alt="Group" /></td>
</tr>
</tbody>
</table>

### 11.4.3 SCEEdge Resolution

SCEEdge can be used to represent an **Association**.

#### 11.4.3.1 Association

Although an Association is placed in the SCE metamodel as a DiagramArtifact, its notation will be rendered with a SCEEdge. When the SCEEdge depicts an Association, its SCEElement MUST be specified.

The following table presents the depiction resolutions for an Association:
Table 33. Depiction Resolution of Association

<table>
<thead>
<tr>
<th>SCE Element</th>
<th>Depiction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association where associationDirection is none.</td>
<td>- - - - - - - - -</td>
</tr>
<tr>
<td>Association where associationDirection is one.</td>
<td>- - - - - - ➔</td>
</tr>
<tr>
<td>Association where associationDirection is both.</td>
<td>← - - - - - - ➔</td>
</tr>
</tbody>
</table>
Annex C: Mapping to BPMN

The elements of SCE are not currently 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. Mapping to/from BPMN Base Element/Root Element

<table>
<thead>
<tr>
<th>BPMN Element/Property</th>
<th>SCE Element/Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseElement</td>
<td>SCEElement</td>
</tr>
<tr>
<td>BaseElement.id</td>
<td>SCEElement.identifier</td>
</tr>
<tr>
<td>Not used in BPMN BaseElement. The name property is included in specific BPMN elements that may have a name.</td>
<td>SCEElement.name</td>
</tr>
<tr>
<td>Not included in BPMN.</td>
<td>SCEElement.aliasID</td>
</tr>
<tr>
<td>Not included in BPMN.</td>
<td>SCEElement.humanID</td>
</tr>
<tr>
<td>RootElement (extends BaseElement with no additional properties)</td>
<td>Not in SCE. SCEElement would be a substitute.</td>
</tr>
</tbody>
</table>

Table 35. Mapping to/from BPMN Definitions

<table>
<thead>
<tr>
<th>BPMN Element/Property</th>
<th>SCE Element/Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definitions</td>
<td>SCEDefinitions</td>
</tr>
<tr>
<td>Definitions.name</td>
<td>See SCEElement.name</td>
</tr>
<tr>
<td>Definitions.targetNamespace</td>
<td>SCEDefinitions.targetNamespace</td>
</tr>
<tr>
<td>Definitions.expressionLanguage</td>
<td>Not in SCE since expressions are not included. This is BPMN specific metadata.</td>
</tr>
<tr>
<td>Definitions.typeLanguage</td>
<td>Not in SCE since expressions are not included. This is BPMN specific metadata.</td>
</tr>
<tr>
<td>Definitions.exporter</td>
<td>SCEDefinitions.exporter</td>
</tr>
<tr>
<td>Definitions.exporterVersion</td>
<td>SCEDefinitions.exporterVersion</td>
</tr>
<tr>
<td>Not included in BPMN</td>
<td>SCEDefinitions.tag</td>
</tr>
<tr>
<td>Not included in BPMN</td>
<td>SCEDefinitions.version</td>
</tr>
<tr>
<td>Not included in BPMN</td>
<td>SCEDefinitions.versionDate</td>
</tr>
</tbody>
</table>

Annex D: Mapping to CMMN

The elements of SCE are not currently 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.  Mapping to/from CMMN CMMNElement**

<table>
<thead>
<tr>
<th>CMMN Element/Property</th>
<th>SCE Element/Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMMNElement</td>
<td>SCEElement</td>
</tr>
<tr>
<td>CMMNElement.id</td>
<td><strong>SCEElement.identifier</strong></td>
</tr>
<tr>
<td>Not used in CMMNElement. The name property is included in specific CMMN elements that may have a name.</td>
<td>SCEElement.name</td>
</tr>
<tr>
<td>Not included in CMMN.</td>
<td>SCEElement.aliasID</td>
</tr>
<tr>
<td>Not included in CMMN.</td>
<td>SCEElement.humanID</td>
</tr>
</tbody>
</table>

**Table 37.  Mapping to/from CMMN Definitions**

<table>
<thead>
<tr>
<th>CMMN Element/Property</th>
<th>SCE Element/Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definitions</td>
<td>SCEDefinitions</td>
</tr>
<tr>
<td>Definitions.name</td>
<td>See SCEElement.name</td>
</tr>
<tr>
<td>Definitions.targetNamespace</td>
<td><strong>SCEDefinitions.targetNamespace</strong></td>
</tr>
<tr>
<td>Definitions.expressionLanguage</td>
<td>Not in <strong>SCE</strong>. This is <strong>CMMN</strong> specific metadata.</td>
</tr>
<tr>
<td>Definitions.exporter</td>
<td><strong>SCEDefinitions.exporter</strong></td>
</tr>
<tr>
<td>Definitions.exporterVersion</td>
<td><strong>SCEDefinitions.exporterVersion</strong></td>
</tr>
<tr>
<td>Definitions.author</td>
<td>Not in <strong>SCE</strong>. This is <strong>CMMN</strong> specific metadata, but could be provided by <strong>PPMN</strong>.</td>
</tr>
<tr>
<td>Definitions.creationDate</td>
<td>Not in <strong>SCE</strong>. This is <strong>CMMN</strong> specific metadata, but could be provided by <strong>PPMN</strong>.</td>
</tr>
<tr>
<td>Not included in CMMN</td>
<td>SCEDefinitions.tag</td>
</tr>
<tr>
<td>Not included in CMMN</td>
<td>SCEDefinitions.version</td>
</tr>
<tr>
<td>Not included in CMMN</td>
<td><strong>SCEDefinitions.versionDate</strong></td>
</tr>
</tbody>
</table>

**Annex E: 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. Mapping to/from DMN DMNElement/NamedElement

<table>
<thead>
<tr>
<th>DMN Element/Property</th>
<th>SCE Element/Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMNElement</td>
<td>SCEElement</td>
</tr>
<tr>
<td>DMNElement.id</td>
<td>SCEElement.identifier</td>
</tr>
<tr>
<td>DMNElement.Description</td>
<td>SCE Documentation.body</td>
</tr>
<tr>
<td>DMNElement.Label</td>
<td>SCE Category.name</td>
</tr>
<tr>
<td>Not used in DMN DMNElement. The name property is included in specific BPMN elements that may have a name.</td>
<td>SCEElement.name</td>
</tr>
<tr>
<td>Not included in DMN.</td>
<td>SCEElement.aliasID</td>
</tr>
<tr>
<td>Not included in DMN.</td>
<td>SCEElement.humanID</td>
</tr>
<tr>
<td>NamedElement (extends DMNElement)</td>
<td>Not in SCE. SCEElement would be a substitute.</td>
</tr>
<tr>
<td>NamedElement.name</td>
<td>SCEElement.name</td>
</tr>
</tbody>
</table>

### Table 39. Mapping to/from DMN Definitions

<table>
<thead>
<tr>
<th>DMN Element/Property</th>
<th>SCE Element/Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definitions</td>
<td>SCEDefinitions</td>
</tr>
<tr>
<td>Definitions.namespace</td>
<td>SCEDefinitions.targetNamespace</td>
</tr>
<tr>
<td>Definitions.expressionLanguage</td>
<td>Not in SCE. This is DMN specific metadata.</td>
</tr>
<tr>
<td>Definitions.typeLanguage</td>
<td>Not in SCE. This is DMN specific metadata.</td>
</tr>
<tr>
<td>Definitions.exporter</td>
<td>SCEDefinitions.exporter</td>
</tr>
<tr>
<td>Definitions.exporterVersion</td>
<td>SCEDefinitions.exporterVersion</td>
</tr>
<tr>
<td>Not included in DMN</td>
<td>SCEDefinitions.tag</td>
</tr>
<tr>
<td>Not included in DMN</td>
<td>SCEDefinitions.version</td>
</tr>
<tr>
<td>Not included in DMN</td>
<td>SCEDefinitions.versionDate</td>
</tr>
</tbody>
</table>