Date: November 2022

Pedigree and Provenance Model and Notation (PPMN)

Beta 1 – v1.0

OMG Document Number: dtc/22-11-05

Standard Document URL: https://www.omg.org/spec/PPMN

This OMG document replaces the submission document (bmi/22-09-03). 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 March 24, 2023.

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 October 2023. 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, 109 Highland Avenue, Needham, MA 02494, 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

Annexes

Table of Figures

Table of Tables

Preface

OMG

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

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

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

OMG Specifications

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

https://www.omg.org/spec

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

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

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

Issues

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

1 Scope

A Pedigree and Provenance Model and Notation (**PPMN**) model is a repository of elements capturing the lineage, custody and/or ownership of entities of interest. PPMN models may include elements representing the history of the entities of interest as well as specifications of expected events and processes (herein referred to generally as "occurrences") related to types of entities of interest.

Following the approach of BPM+ Knowledge Package Model and Notation (**BKPMN)** and Shared Data Model and Notation (**SDMN)**, **PPMN** is structured to be dependent on the elements defined in Specification Common Elements (**SCE** [OMG doc number bmi-2021-12-09]**)**. Other Business Modeling and Integration (BMI) Task Force and Healthcare Domain Task Force (HDTF) specifications may also utilize the elements of **SCE** as those specifications are updated in the future.

2 Conformance

2.1 General

Software can claim compliance or conformance with **PPMN 1.0** if, and only if, the software fully matches the applicable compliance points as stated in the specification. In addition, the structural elements provided by Specification Common Elements (**SCE) 1.0** [OMG doc number bmi-2021-12-09]) are also required in a compliant or conformant software solution. 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.

2.2 PPMN Modeling Conformance

The implementation claiming conformance to the Pedigree and Provenance Model and Notation SHALL comply with all of the requirements set forth in Clauses 8, 9, 10, 11, 12, 13, and 14; and it SHALL be conformant with the Visual Conformance in Clause 2.3.

This compliance point is intended to be used by **PPMN** modeling tools.

2.3 Visual Conformance

An implementation that creates and displays **PPMN** models SHALL conform to the specifications and restrictions with respect to diagrammatic relationships between graphical elements, as described in Clause 14. A key element of **PPMN** is the choice of shapes and icons used for the graphical elements identified in this specification. The intent is to create a standard visual language that all PPMN modelers will recognize and understand. An implementation that creates and displays **PPMN** models SHALL use the graphical elements, shapes, markers and decorators illustrated in this specification.

There is flexibility in the size, color, line style, and text positions of the defined graphical elements, except where otherwise specified. In particular:

- **PPMN** 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).
- Graphical elements, shapes, and decorators MAY be of any size that suits the purposes of the modeler or modeling tool with the condition that the additional graphical elements SHALL NOT conflict with any current BPM+ Standard defined graphical element.
- 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 SHALL NOT conflict with any current BPM+ Standard defined line style.

The following extensions to a **PPMN** model are permitted:

- New decorators or indicators MAY be added to the specified graphical elements. These decorators or indicators could be used to highlight a specific attribute of a **PPMN** element or to represent a new subtype of the corresponding concept with the condition that the additional graphical elements SHALL NOT conflict with any current BPM+ Standard defined decorator or indicator.
- A new shape representing a new kind of **PPMN** element MAY be added to a model with the condition that the shape SHALL NOT conflict with the shape specified for any other BPM+ Standard element or decorator.
- Graphical elements MAY be colored, and the coloring MAY have specified semantics that extend the information conveyed by the element as specified in this standard.
- The line style of a graphical element MAY be changed, but that change SHALL NOT conflict with any other line style REQUIRED by this specification or the other BPM+ Standards.
- An extension SHALL NOT change the specified shape of a defined graphical element or decorator. (e.g., changing a square into a triangle, or changing rounded corners into squared corners, etc.).

This compliance point is intended to be used by entry-level **PPMN** tools.

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
- [BPMN] OMG Business Process and Model Notation (BPMN™): https://www.omg.org/bpmn/
- [CMMN] OMG Case Management Model and Model Notation (CMMN™): https://www.omg.org/spec/CMMN/
- [DD] Diagram Definition (DD™)
- [DMN] OMG Decision Model and Model Notation (DMN™): https://www.omg.org/spec/DMN/
- [MOF] Meta Object Facility (MOF™): https://www.omg.org/spec/MOF/
- [SCE] Specification Core Elements (SCE): https://www.omg.org/spec/SDMN/
- [UML] Unified Modeling Language ™ (UML®): https://www.omg.org/spec/UML
- [XMI] XML Metadata Interchange (XMI®) https://www.omg.org/spec/XMI

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.

- [MDMI] OMG Model Driven Message Interoperability (MDMI), Version 1.0: https://www.omg.org/spec/MDMI/
- [SysML] OMG Systems Modeling Language (SysML®): https://www.omg.org/spec/SysML/

4 Terms and Definitions

The table below presents a glossary for this specification:

Table 1. Glossary

Term	Definition
Area	A kind of location that encompasses some region in the world.
Chain of Control	The succession of controllers of an entity of interest. Also known as Chain of Custody.
Chain of Custody	The succession of custodians of an entity of interest. Also known as Chain of Control.
Chain of Ownership	The succession of owners of an entity of interest.
Channel	The "route" by which an entity of interest was obtained.
Controller	The party that holds an entity of interest for the owner. Also known as Custodian.
Custodian	The party that holds an entity of interest for the owner. Also known as Controller.
Delegation	A kind of Position Assignment relationship that states that one Party has been assigned a set of responsibilities by some authority.
Entity	An individual concept or informational or physical artifact that is concretized in digital or other media or in a physical representation. The W3C PROV-DM defines an entity as "*An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.*"[1]
Entity of Interest	The Entity (e.g., artifact, document, record, collection of materials or data element) whose provenance or pedigree is being recorded.
Geospatial Extent	A location that is a volume in the world such as a container or a room.
Location	A particular place or position.
Network Address	The address of an element or node on a network.
Non-Human Agent	Some type of automated system.
Occurrence	A "happening" of importance in a domain in some context.
Organization	Organization is used to represent a group of Parties. The group may be a company, a department within a company, a club, a consortium, or some other group.
Organization Structure Relationship	A kind of Party Relationship used to indicate internal structural relationships of a Party.
Owner	The Party that owns an entity as property. Merriam-Webster: a person who owns something : one who has the legal or rightful title to something : one to whom property belongs.
Ownership	The state, relation, or fact of being an owner. (Merriam-Webster)
Party	An abstract concept representing a Person, Role, Organization, or other entity involved in some activity, interaction or endeavor.
Party Relationship	A kind of relationship that exists between two Parties.
Party Role	A role played by a Party in some context. For instance, a Buyer or a Supplier.

[1] https://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-entity

Path	An ordered collection of Locations.
Pedigree	Pedigree captures the "lineage" of an entity of interest. In other words, the pedigree of an Entity of Interest is the lattice formed by the sequence of activities, processes, and/or derivations performed on other entities (a.k.a, its "ancestors"), the inputs to those activities, processes, and/or derivations, and their outputs that result in or produce the Entity of Interest.
Pedigree Chain	A succession of events that have occurred in the life of an entity of interest with respect to a particular interested party.
Person	An individual homo sapiens.
Physical Address	A physical location in the real world that has an identifiable address.
Position	A Position is a formally defined role in an Organization filled by some Person. Positions are often associated with a set of responsibilities in some context. Examples of Positions include Chief Executive Officer or Technical Staff Member.
Position Assignment	Position Assignment indicates a Party is assigned to a particular Position for a particular period of time.
Provenance	Provenance captures the chain of custody or chain of ownership of an entity of interest.
Space-Time	A Location at a particular point in time.

5 Symbols

6 Additional Information

6.1 Conventions

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

6.2 Typographical and Linguistic Conventions and Style

This document incorporates the following conventions:

- The keywords “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document are to be interpreted as described in RFC-2119.
- A **term** is a word or phrase that has a special meaning. When a term is defined, the term name is highlighted in **bold** typeface.
- A reference to another definition, section, or specification is highlighted with underlined typeface and provides a link to the relevant location in this specification.
- A reference to a graphical element is highlighted with a bold, capitalized word (e.g., **ProcessRef**).
- A reference to a non-graphical element or **PPMN**, **Parties**, or **SCE** concept is highlighted by being italicized (e.g., *Entity*).
- 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:

- [1] — exactly once
- [0..1] — 0 or 1
- [0..*] — 0 or more
- [1..*] — 1 or more

- Attributes separated by | and grouped within { and } — alternative values
  - <value> — default value
  - <type> — the type of the attribute

6.3 Display of Metamodel Diagrams

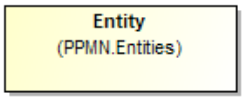
The metamodel presented in these sections utilizes the patterns and mechanisms that are used for the current **BPM+** specifications. **BPM+** specifications rarely display the entire metamodel of a technical specification in a single diagram. The entire metamodel would be very large, complicated, and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

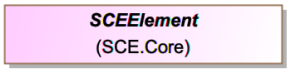
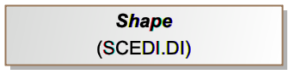
The metamodel diagrams are Unified Modeling Language (UML) structure diagrams. In addition to the metamodel, OMG specifications provide XML schemas which map to the metamodels. In general, it is through XML documents that **BPM+** models are stored and exchanged.

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

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

Table 2. PPMN Metamodel Color-Coding

Element	Description	Example Color
PPMN Class	These elements include the namespace in the model of the element in parenthases below the element name when that element is outside the namespace of the current diagram. These elements are color-coded light yellow and the border line color is black (see figure to the right). These make up the majority of metamodel elements shown in this specification.	
Parties General Class	These elements include the namespace in the model of the element in parenthases below the element name when that element is outside the namespace of the current diagram. These elements are color-coded light green and the border line color is black (see figure to the right). These elements are primarily found in the Parties Model section of this specification but are also shown in the Pedigree and Provenance Model and Notation section of this specification.	

SCE Class	Metamodel elements from the **SCE 1.0** specification [OMG doc number bmi-2021-12-09] are shown in **PPMN** metamodel diagrams when **PPMN** or **Parties Model** elements are dependent on a **SCE** element. These elements include the namespace in the metamodel in parenthases below the element name and these elements are color-coded lavender (see figure to the right).	
External Class	Classes from specifications that are not specifically part of the BPM+ stack of standards can be included in metamodel diagrams and display the owner of the language in parenthases below the element name and these elements are color-coded light-gray. (see figure to the right).	

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 SHALL NOT conflict with any current defined line style of the diagram.

6.5 Abbreviations

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

Table 3. Acronyms

Acronym	Definition
BHMN	BPM+ Harmonization Model and Notation
BKPMN	BPM+ Knowledge Package Model and Notation
BPM+	Business Process Management Plus
BPMN	Business Process Model and Notation
CMMN	Case Management Model and Notation
DC	Diagram Commons
DD	Diagram Definition
DI	Diagram Interchange
DMN	Decision Model and Notation
MDMI	Model Driven Message Interoperability
MOF	Meta Object Facility
OMG	Object Management Group

PPMN	Provenance and Pedigree Model and Notation
RFC	Remote Function Call
SCE	Specification Common Elements
SDMNDI	Shared Data Model and Notation Diagram Interchange
SDMN	Shared Data Model and Notation
SysML	Systems Modeling Language
URI	Uniform Resource Identifier
XMI	XML Metadata Interchange
XML	Extensible Markup Language

6.6 Structure of this Document

This document provides a brief introduction to **SDMN** and its purpose (see the section entitled "Overview"). The introduction is followed by normative clauses that define the elements of the specification and their properties and associations (see the sections entitled "SDMN Metamodel" (Clause 9); "SDMN Model Elements" (Clause 10); "SDMN Models" (Clause 11); "SDMN Library" (Clause 12); "Mapping to BPM+ Models" (Clause 13); and "SDMN Diagram Interchange" (Clause 16)).

UPDATE

6.7 Acknowledgements

Supporting Organizations

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

Department of Veterans Affairs
cébé IT
Knowledge Management LLC
Thematix Partners LLC.

Special Acknowledgements

The following individuals provided major input to this specification:

John Butler
Claude Baudoin
Thomas Beale
Elisa Kendall
Robert Lario
Pete Rivett
Evan Wallace
Steve White

7 Overview

The goal of the Pedigree and Provenance Model and Notation specification is to provide a common language for expressing information about the origin, evolution, ownership, custody and potential end of life of entities of interest. The primary conceptual elements in the PPMN language are Entities (the items of interest), Occurrences (events that affect an Entity) and Parties (responsible actors).

The **PPMN** specification is organized into a number of packages that together comprise the full model for expressing the pedigree and provenance of entities of interest. Starting at the bottom of the figure below, **PPMN**

uses elements from the **SCE** model as the basis of its elements. All elements in **PPMN** are specializations of **SCE** *BaseElement* directly or *NamedElement*.

PPMN also uses elements from the **Parties** Model as shown in the second layer from the bottom. These elements support the specification of various types of parties including organizations, people, positions and roles. **Parties** also defines *PartyTypes.* As described in the sections below, *PartyTypes* provide the ability to state what kind of *Party* is expected to play some role within an *Occurrence*.

The next layer up contains the basic **PPMN** elements on which the rest of the specification is built – *Entities* and *Occurrences*. Entities are the things of interest from a pedigree and provenance perspective. Occurrences are the "things that happen" related to these entities and parties. The layer also includes Rationale – a set of model elements supporting the capture of the basis or reason for an Occurrence or OccurrenceType.

The fourth layer comprises a set of packages that include elements used to elaborate *Entities*, *Occurrences*, and *Parties* from a pedigree and provenance perspective. *Delegation* includes elements that support the delegation of responsibilities from one *Party* to another. The *Additional Relationships* package includes several specialized relationships of use in capturing pedigree and provenance.

The fifth layer comprises the pedigree and provenance specific elements as well as mechanisms to extend the model. The *Pedigree* and *Provenance* packages use elements from the lower four layers to provide the specific metadata to track pedigree, the lineage of entities of interest, and provenance, the ownership and custody of those entities. The *Extensions* package provides the ability to add custom metadata either through annotations or adornments. *Claims* are mechanisms that support the ability to capture who made a particular statement about an Occurrence and whether the statement was intended to indicate that the Occurrence did in fact happen, did not happen, or may have happened.

Finally, the Packaging package provides elements necessary to bundle pedigree and provenance occurrence instances and types into coherent sets either for storage or for exchange.

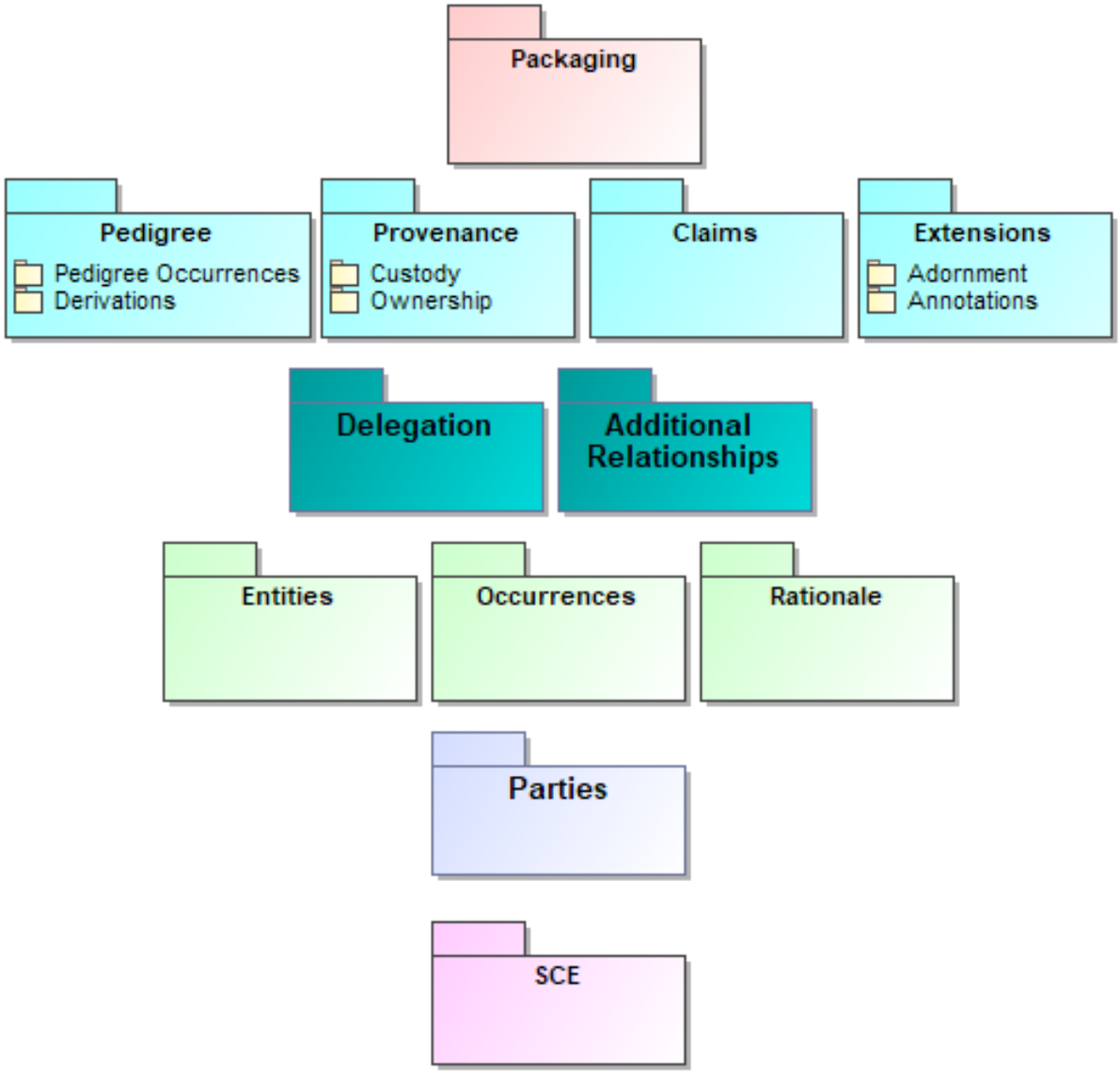


Figure 1: PPMN Packaging Overview

8 Pedigree and Provenance Model and Notation

PPMN is comprised of a number of packages that group closely related elements in particular subdomains. As shown in the figure below, these domains build from the common elements specified in the Specification Common Elements (**SCE**) package. **PPMN** incorporates additional basic elements and primitives that form the foundation of the rest of the model.

On top of these basic elements **PPMN** lays further foundation in the form of Parties, Entities and Occurrences. Parties support the specification of the organizations, people and roles they play. Entities support identifying complex "things of interest" as well as references to things that might be external to the system containing the pedigree and provenance metadata. Occurrences provide a general mechanism for identifying *things that happen* in the form of general graphs.

Pedigree, Provenance and Extensions are built on top of the packages described above. Both Pedigree and Provenance extend occurrences related to entities show the parties involved in those occurrences. Pedigree occurrences describe the creation and/or evolution of entities while provenance occurrences describe the ownership and/or custody of entities. The Extensions package provides mechanisms for adding metadata to other elements of the model.

Finally, **PPMN** Packaging provides mechanisms for packaging occurrences and occurrence types (expected occurrences) for exchange or other purposes.

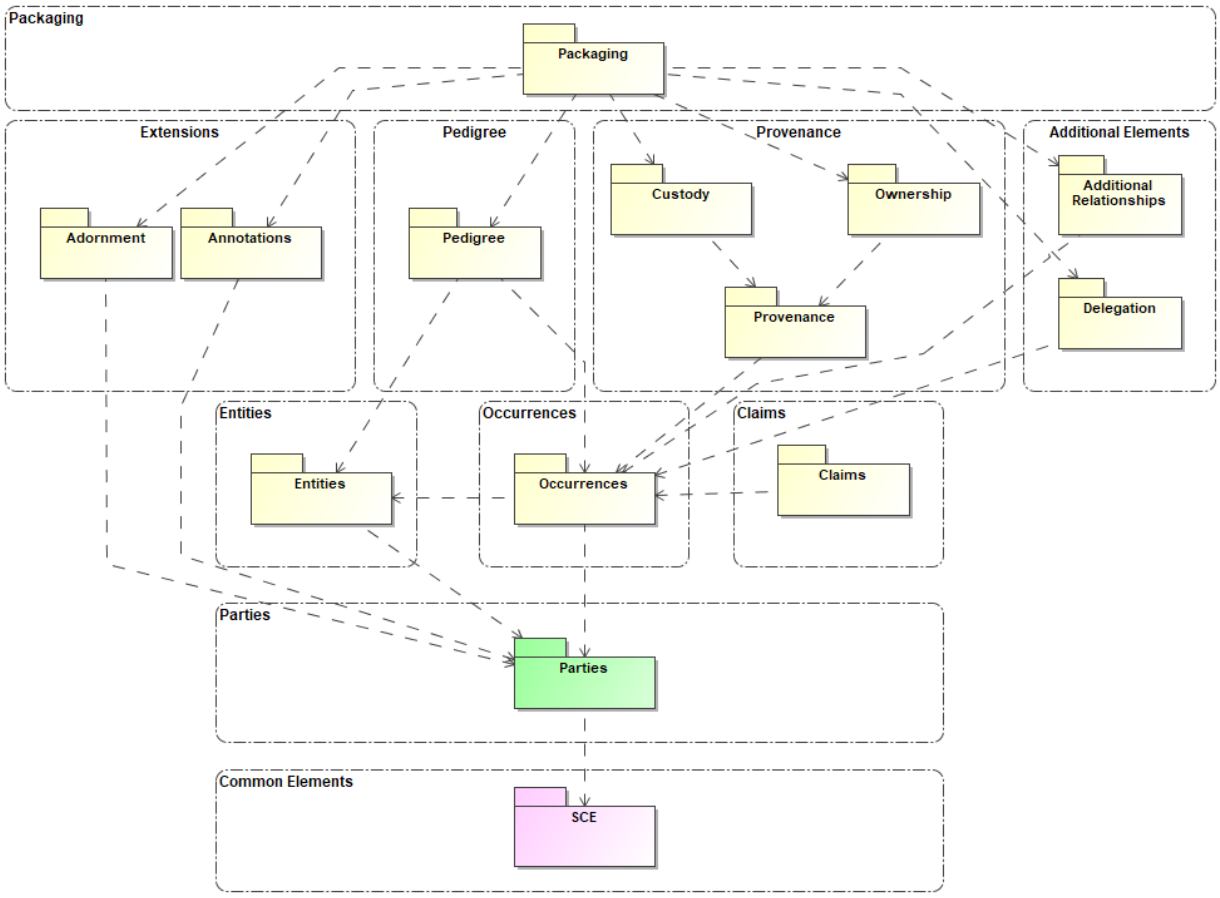


Figure 2: Pedigree and Provenance Packaging

8.1 Entities

PPMN is concerned with recording relevant information about things of interest to stakeholders. The Entities package contains elements that represent those (potentially complex) things that are of interest from a pedigree and/or provenance perspective.

Entities are concepts or objects that may have a physical or digital embodiment. Entities may be of some defined type, *EntityType*, with a defined format and reside at some location. Entities may represent some other thing of interest through the `entityURI` property. All *Entity*-related classes are ultimately *SCEElements* and as such have a `name`, `id`, and `URI`. Entities may also comprise other Entities using the *EntityComposition* relationship.

EntitySnapshots represent some entity at a particular point in time. Like Entities, they may comprise other Entities using the EntityComposition relationship.

EntityTypes, as with *Entities*, have snapshots (*EntityTypeSnapshots*) and can comprise other *EntityTypes* through *EntityTypeComposition.* As with *EntityComposition, EntityTypeComposition* is also a specialization of *ElementRelationship.*

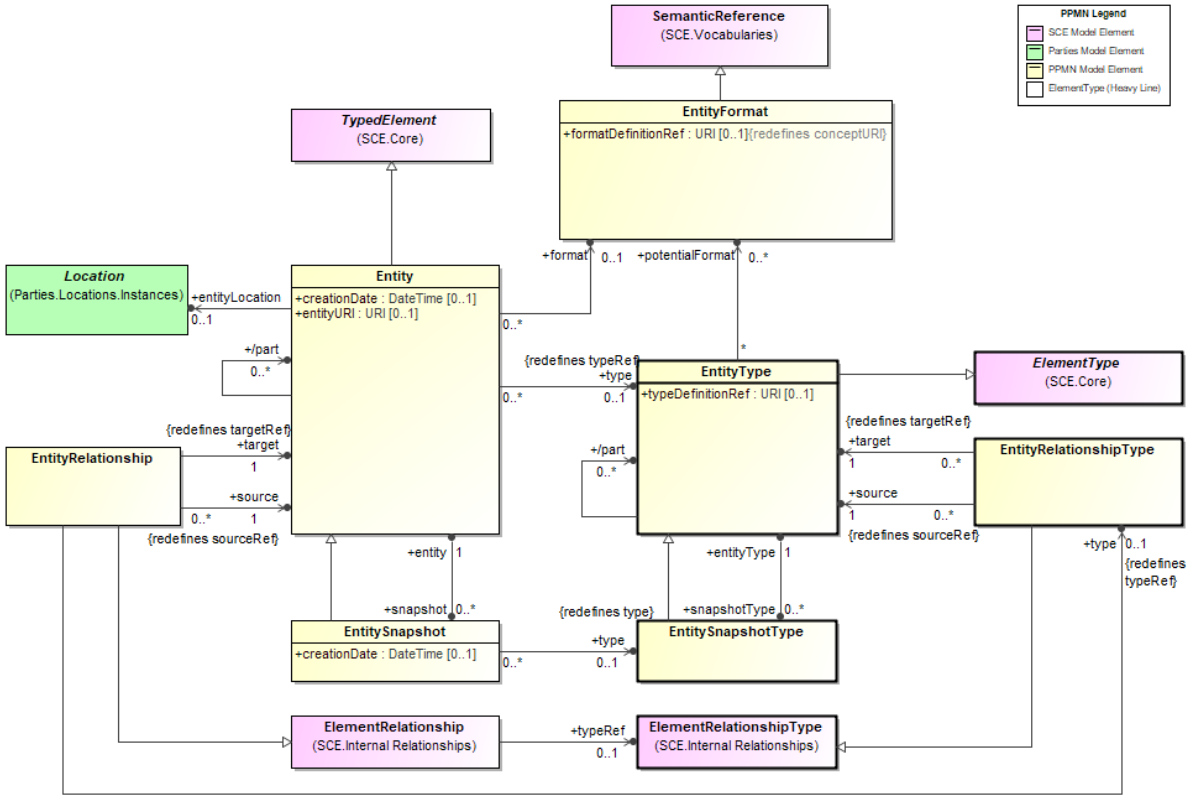


Figure 3: Entities and EntityTypes

8.1.1 Entity

An individual concept or informational or physical artifact that is concretized in digital or other media or in a physical representation. The W3C PROV-DM defines an entity as "*An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.*"[2] Entities may have a type and format, captured through the *EntityType* and *EntityFormat*, respectively. These two classes are used together to support specifying generally what kind of thing an *Entity* is and the form it may take. For example, the *EntityType* might be a "building permit" and the *EntityFormat* might be ".gif". Additionally, Entities may have a location as captured by the `entityLocation` property.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

[2] https://www.w3.org/TR/2013/REC-prov-dm-20130430/#term-entity

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

Table 4. Entity Attributes and/or Associations

Property/Association	Description
creationDate : DateTime [0..1]	The date the *Entity* was created.
entityLocation : Location [0..1]	The location of the *Entity*.
entityURI : URI [0..1]	A URI to the Entity.
format : EntityFormat [0..1]	The format of the *Entity.*
part : Entity [0..*]	The *Entity* or *Entities* that is/are contained by the *Entity*.
snapshot : EntitySnapshot [0..*]	The snapshots of the *Entity* that represent the *Entity* at some particular point in time, particular *Location*, or both.
type : EntityType [0..1]	The type of the *Entity*.

8.1.2 EntityFormat

A kind of *SemanticReference* that represents the format of an *Entity*. It can be something as simple as "mime types" or the specification of a format documented in a formal format registry.

Generalizations

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

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

Properties

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

Table 5. EntityFormat Attributes and/or Associations

Property/Association	Description
formatDefinitionRef : URI [0..1]	The identifier of the format within the specified format registry. For example "dicom" if the registry is that of W3C mime types. This is not the usual "id" found commonly in this specification. This is a "stringified" (if necessary) unique id in the context of the .formatRegistry.

8.1.3 EntityRelationship

A kind of *ElementRelationship* that represents an expected relationship between two *Entities.* The kind of *EntityRelationship* is specified by the `type` property inherited from *ElementRelationship*.

Generalizations

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

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

Properties

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

Table 6. EntityRelationship Attributes and/or Associations

Property/Association	Description
occurrence : ActivityOccurrence [0..1]	The *Occurrence* that resulted in the relationship.
source : Entity [1]	The source *Entity* of the relationship.
target : Entity [1]	The target *Entity* of the relationship.
type : EntityRelationshipType [0..1]	A specification of the type of EntityRelationship.

8.1.4 EntityRelationshipType

A kind of *ElementRelationship* that represents an expected relationship between two *EntityTypes.* The kind of *EntityTypeRelationship* is specified by the `type` property inherited from *ElementRelationship*.

Generalizations

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

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

Properties

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

Table 7. EntityRelationshipType Attributes and/or Associations

Property/Association	Description
source : EntityType [1]	The source *EntityType* of the relationship.
target : EntityType [1]	The target *EntityType* of the relationship.

8.1.5 EntitySnapshot

A kind of *Entity* that represents a snapshot of another *Entity* at a particular point in time, a particular *Location*, or both. Additionally, *EntitySnapshots* may conain other *Entities* as specified by the `parts` that are captured through the *EntityComposition* relationship.

Generalizations

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

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

Properties

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

Table 8. EntitySnapshot Attributes and/or Associations

Property/Association	Description
creationDate : DateTime [0..1]	The date the *EntitySnapshot* was created.
entity : Entity [1]	The *Entity* that the *EntitySnapshot* represents at some particular point in time and potentially some *Location*.
type : EntitySnapshotType [0..1]	The type of the *Entity*.

8.1.6 EntitySnapshotType

A kind of *EntityType* that represents a expected snapshot of an *EntityType* at a particular point in time, a particular *Location*, or both. Additionally, *EntityTypeSnapshots* may contain other *EntityTypes* as specified by the `part` property that are captured through the *EntityTypeComposition* relationship.

Generalizations

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

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

Properties

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

Table 9. EntitySnapshotType Attributes and/or Associations

Property/Association	Description
entityType : EntityType [1]	The *EntityType* that the *EntityTypeSnapshot* represents at some particular point in time, particular *Location*, or both.

8.1.7 EntityType

EntityType is a designation defined for the convenience of an organization and can be used to define any concept concerning an *Entity* that serves the organization. *EntityType* has 1..* potential formats specified through the `potentialFormat` property to *EntityFormat.* E.g., an *EntityType* might be "Building Layout" and the possible formats may be .gif, .jpeg, or paper.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

Table 10. EntityType Attributes and/or Associations

Property/Association	Description
part : EntityType [0..*]	The *EntityType* or *EntityTypes* that is/are contained by the *EntityType*.
potentialFormat : EntityFormat [0..*]	Formats in which *Entities* of type *EntityType* may exist.
snapshotType : EntitySnapshotType [0..*]	The snapshots of the *EntityType* that represent the *EntityType* at some particular point in time, particular *Location*, or both.
typeDefinitionRef : URI [0..1]	An external definition of the EntityType.

8.2 Occurrences

The Occurrences package contains general elements related to the "happenings" or events that occur over the lifetime of an entity of interest. These happenings might signify anything of interest to some *Party* but are intended capture common properties of pedigree- and provenance-related events.

PPMN *Occurrences* are "happenings" related to one or more *Entities* that have to do with the pedigree or provenance of the *Entity or Entities*. *Occurrences* are *TypedElements* whose type is an *OccurrenceType*. *Occurrences* have a `start` and `end` Date/Time and may occur at some particular `location`. *OccurrenceChains* track some series of *Occurrences* related to some set of *Entities* that are the `subject` of the *Occurrences*.

Occurrences may have a number of different kinds of relationships with other types of elements. These elements include *OccurrenceRelationships*, *OccurrenceDependencies*, and *OccurrenceRoles*. These are all kinds of *ElementRelationship*. *OccurrenceRelationships* track the predecessor *Occurrences* of a particular *Occurrence*. *OccurrenceDependencies* track the *Entities* related to a particular *Occurrence* as well as the role that the *Entity* played in the *Occurrence*. *OccurrenceRoles* capture the role played by *Parties* in the *Occurrence*.

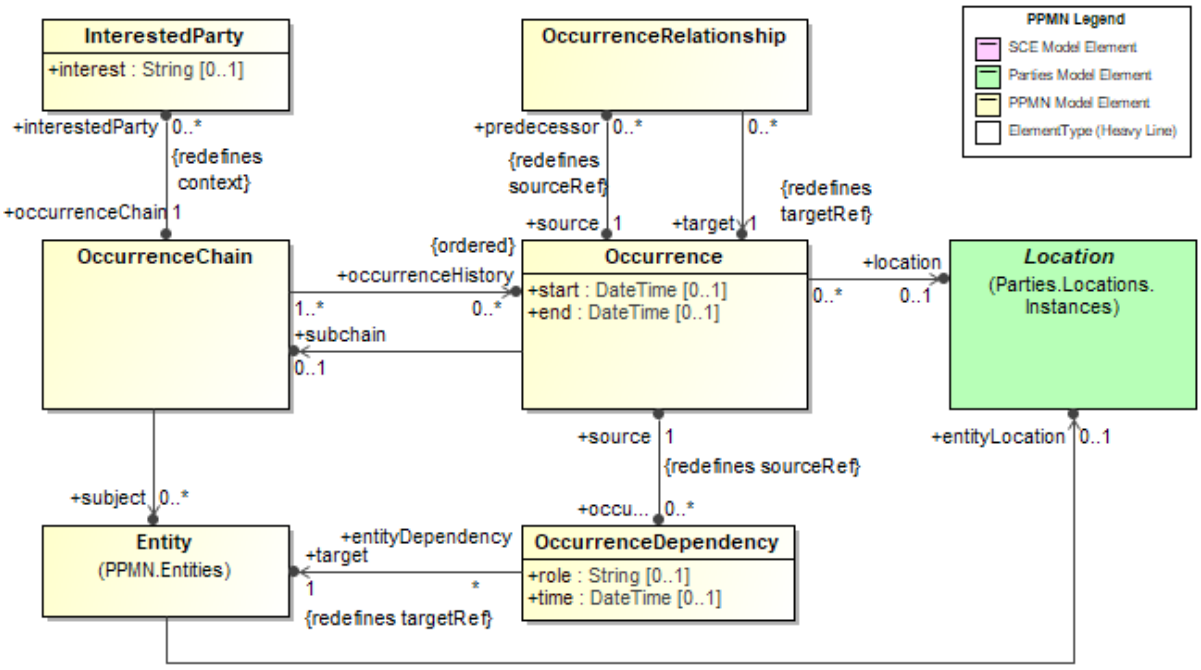


Figure 4: Occurrences - Simplified

In addition, *Occurrences* may also result in some number of *ElementRelationships* between elements that were involved in the *Occurrence*. These include *DerivedFrom* relationships (see section 8.3.2, below) as well as *ResponsibilityRelationships* (see section 8.4.9, below).

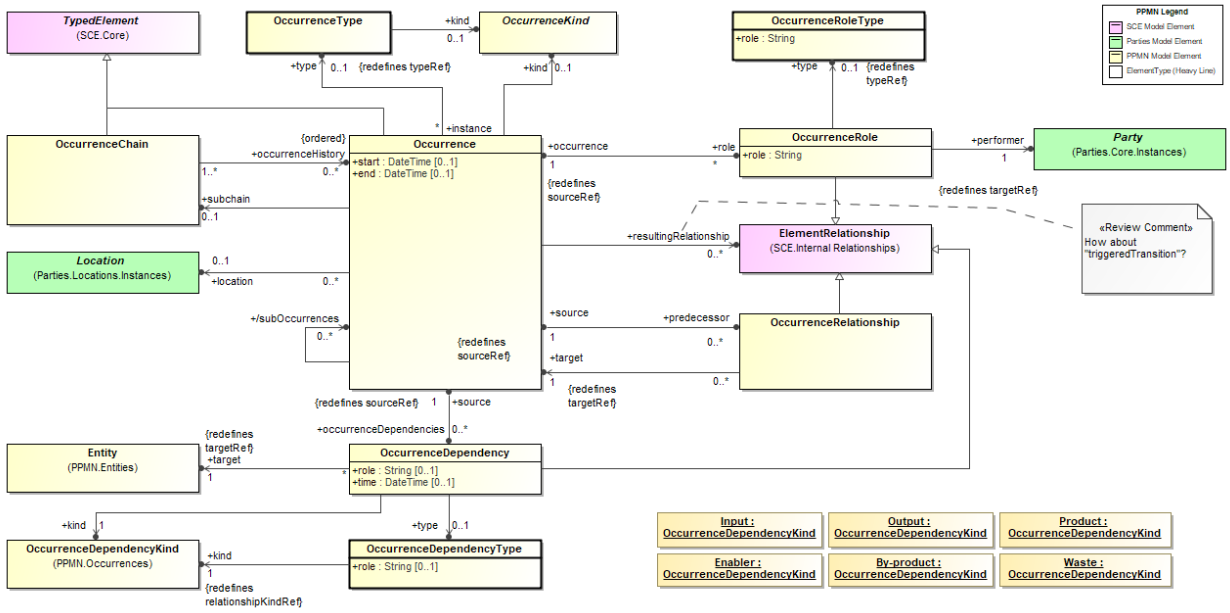


Figure 5: Occurrences

OccurrenceChains are *TypedElements* that track some series of *Occurrences* related to one or more *Entities* that acts as the context of the *Occurrences*.

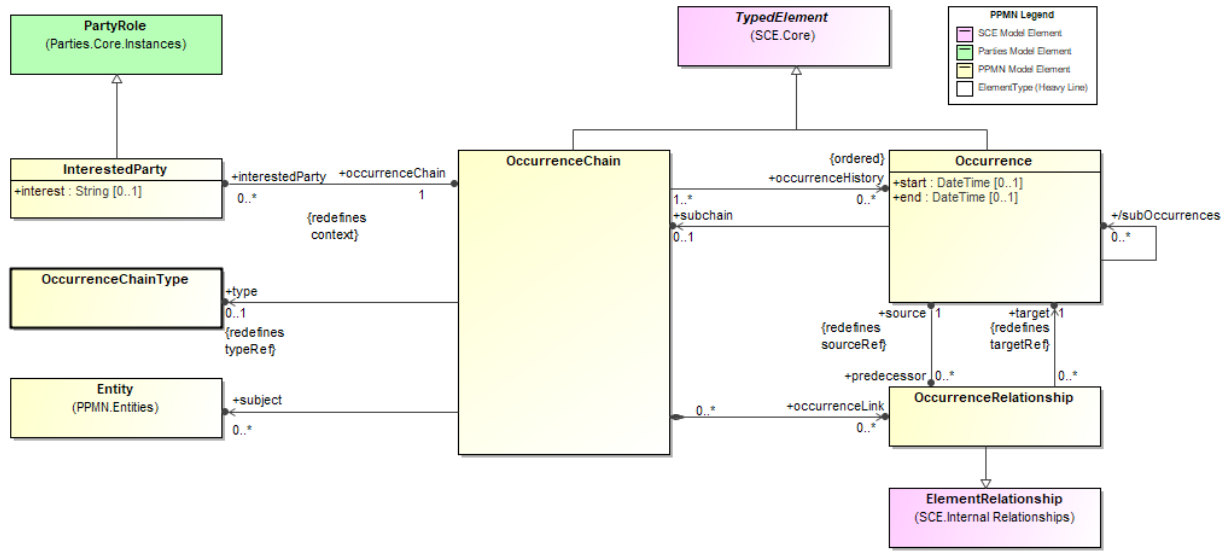


Figure 6: Occurrence Chains

OccurrenceTypes support the definition of expected Occurrences in an *OccurrenceChain*. Essentially, *OccurrenceTypes* represent *Occurrence* instances that are expected to happen to entities of a particular type from the perspective of the *InterestedParties. OccurrenceTypes* can be organized into graphs, *OccurrenceTypeGraphs*, that show an expected sequence or "chain" of those types of *Occurrences*. Further, *OccurrenceTypes* can optionally have sub-chain types so that *OccurrenceTypeGraphs* can be nested within one another. *OccurrenceTypeRole* captures roles expected to be played by *Parties* in those *Occurrences*.

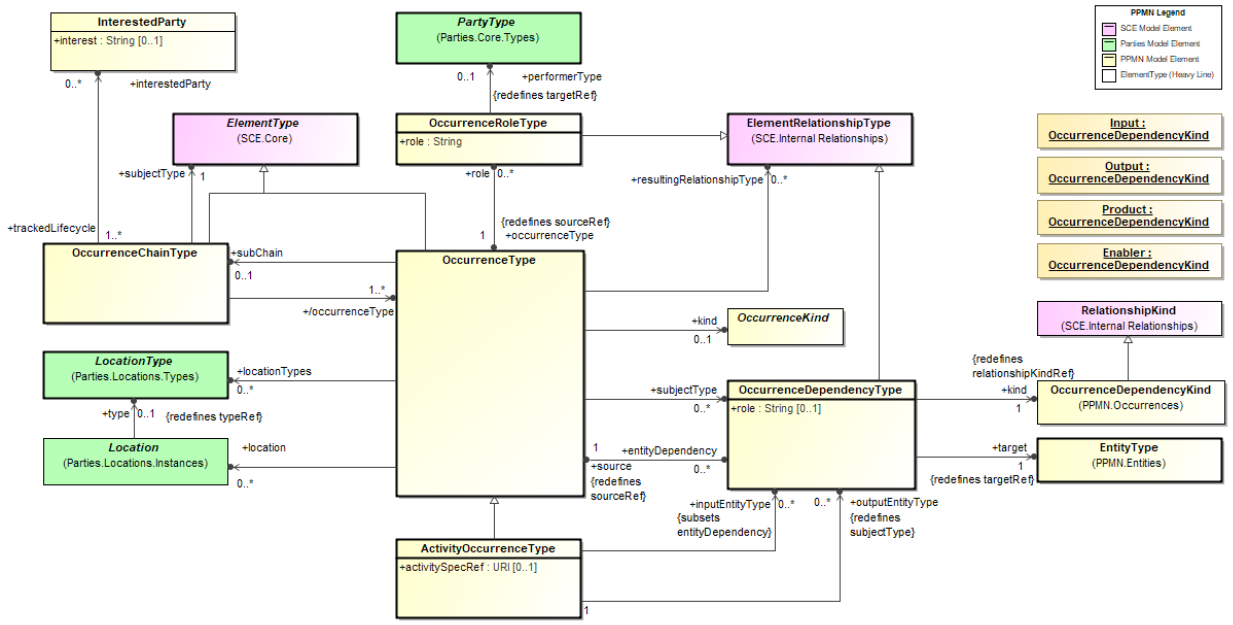


Figure 7: Occurrence Types

Expected *OccurrenceTypes* can be organized into graphs, *OccurrenceTypeGraphs*, that show an expected sequence or "chain" of those types of *Occurrences*. Further, *OccurrenceTypes* can optionally have sub-chain types so that *OccurrenceTypeGraphs* can be nested within one another. *OccurrenceTypeRole* captures roles expected to be played by *Parties*.

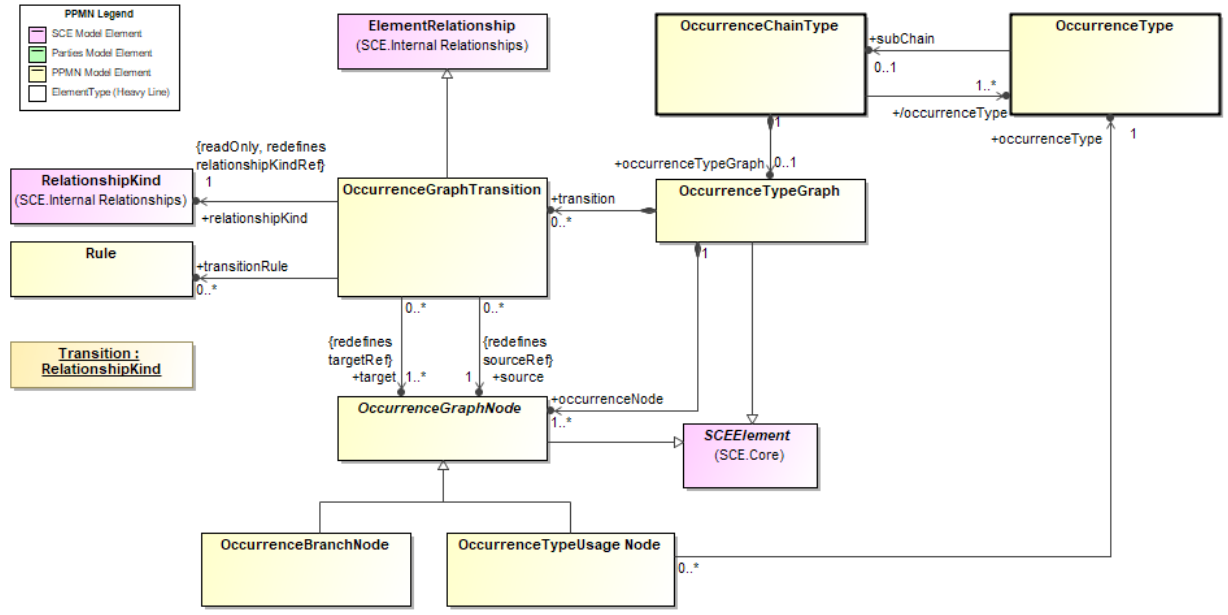


Figure 8: Occurrence Type Graphs

PPMN establishes a pattern of elements that supports the "nesting" of *OccurrenceChains* within an *Occurrence*. This pattern allows for encapsulation of parts of a chain where the details of the *Occurrences* of that part of a larger chain are either not known initially or are not deemed important in some context. The figure below illustrates the this pattern at the “type” level.

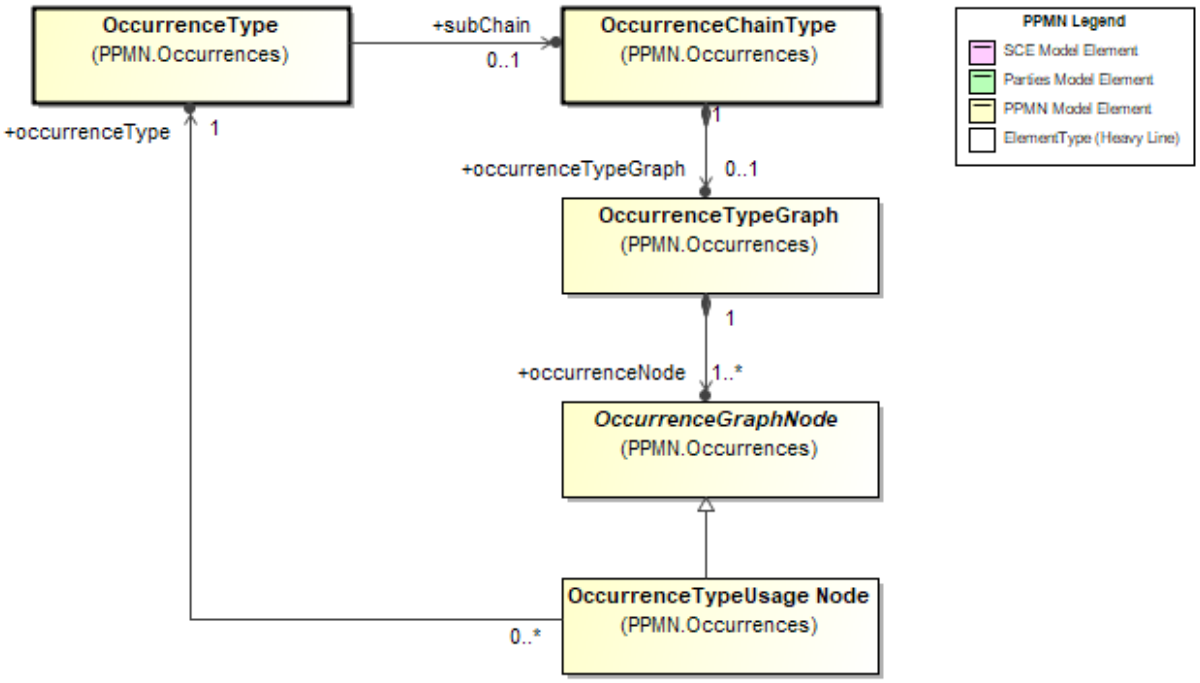


Figure 9: Occurrences Type Pattern

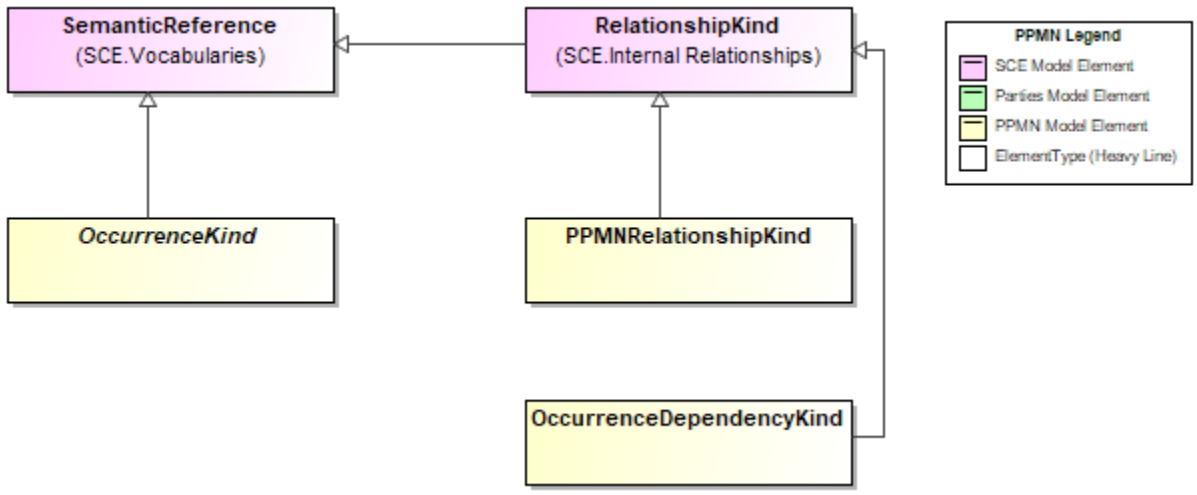


Figure 10: Occurrence Kinds

An *ActivityOccurrence* is a kind of *Occurrence* that represents some activity that produces or modifies one or more entities. The *ActivityType* specifies the type of activity of the *ActivityOccurrence* providing a URI for a specification of the *ActivityType*.

ActivityOccurrence has a name (inherited), a URI reference to a specification of the instance if one exists, and a description. *ActivityOccurrence* includes zero or more references to *Parties* that play a part in the activity through the inherited *OccurrenceRole* property and references to the entities used in the activity through the `inputEntity` property which holds a collection of *OccurrenceDependencies*. Output entities of the activity are captured through the `outputEntities` property.

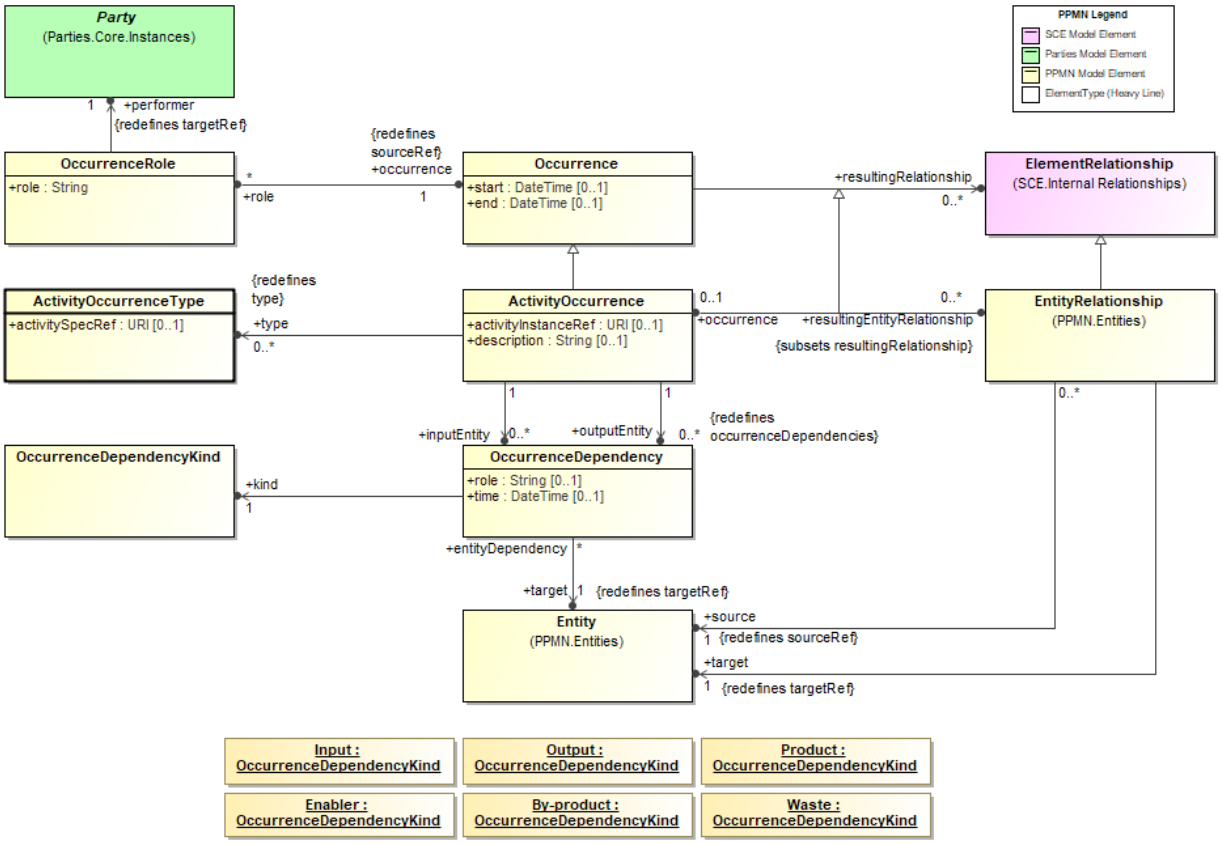


Figure 11: Activity Occurrences

8.2.1 ActivityOccurrence

A kind of *Occurrence* that records the input and output entities of interest as the result of some activity or derivation.

An *ActivityOccurrence* is a kind of *Occurrence* that represents some activity that produces or modifies one or more entities. The *ActivityType* specifies the type of activity of the *ActivityOccurrence* providing a URI for a specification.

ActivityOccurrences have a name (inherited), a URI reference to an instance if one exists, and a description. *ActivityOccurrences* include references to *Parties* that play a part in the activity through the inherited *OccurrenceRole* property and references to the entities used in the activity through the `inputEntity` property which holds a collection of *OccurrenceDependencies*. Output entities of the activity are captured through the `outputEntities` property of *PedigreeOccurrence*.

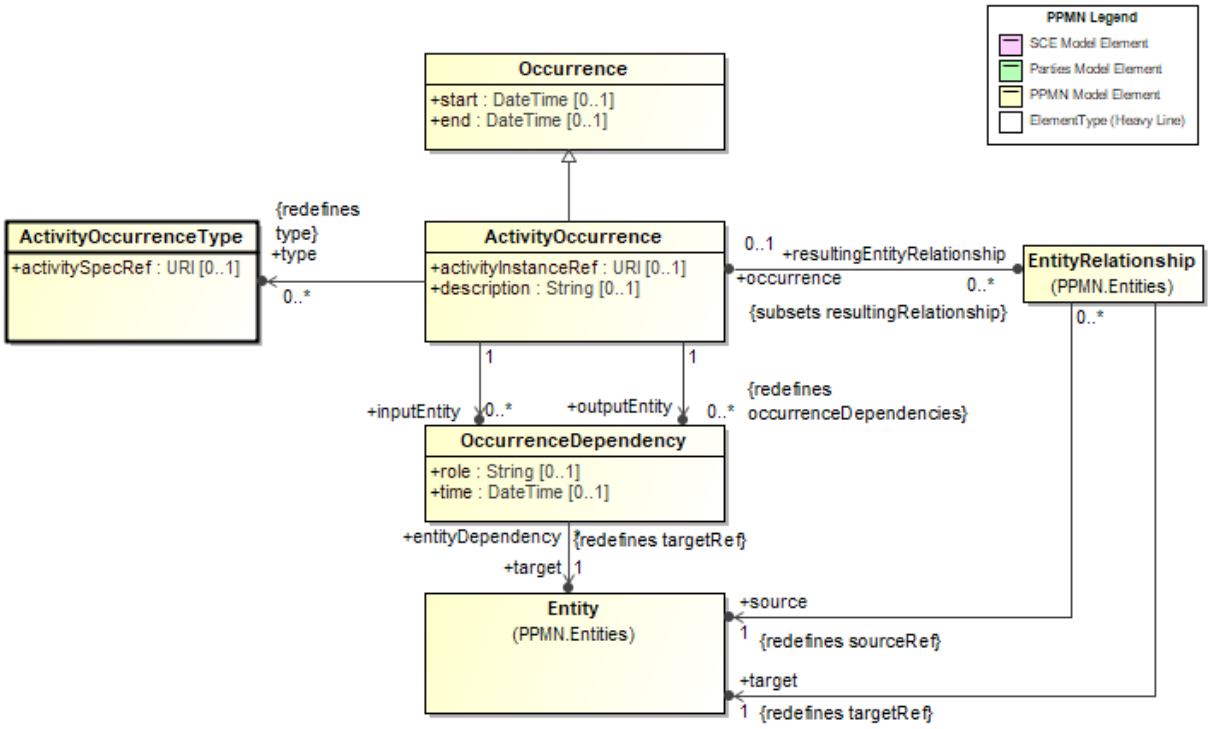


Figure 12: Activity Occurrence

Generalizations

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

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

Properties

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

Table 11. ActivityOccurrence Attributes and/or Associations

Property/Association	Description
activityInstanceRef : URI [0..1]	A reference to an instance that the *ActivityOccurrence* represents. This could be an instance running in a business process execution engine or some other tool.
description : String [0..1]	A textual description of the activity.
inputEntity : OccurrenceDependency [0..*]	A set of dependencies to the entities that were inputs to the *ActivityOccurrence*.
outputEntity : OccurrenceDependency [0..*]	A set of dependencies on entities that were outputs or results of the *ActivityOccurrence*.
resultingEntityRelationship : EntityRelationship [0..*]	EntityRelationships created as a result of the Occurrence.
type : ActivityOccurrenceType [0..*]	The type of the *ActivityOccurrence*.

8.2.2 ActivityOccurrenceType

A potentially complex *OccurrenceType* that identifies an expected activity that may have input and output entities of interest.

Generalizations

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

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

Properties

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

Table 12. ActivityOccurrenceType Attributes and/or Associations

Property/Association	Description
activitySpecRef : URI [0..1]	A reference to a specification for the activity.
inputEntityType : OccurrenceDependencyType [0..*]	A set of dependencies that point to the types of entities that are expected to be consumed or used by instances of the *OccurrenceType.*
outputEntityType : OccurrenceDependencyType [0..*]	A set of dependencies that point to the types of entities that are expected to be produced by instances of the *OccurrenceType.*

8.2.3 InterestedParty

A kind of *PartyRole* that captures the fact that a *Party* has some interest in a particular occurrence chain as specified by its `occurenceChain` property or so some set of *OccurrenceChains* as defined by an *OccurrenceChainType*.

Generalizations

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

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

Properties

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

Table 13. InterestedParty Attributes and/or Associations

Property/Association	Description
interest : String [0..1]	A textual description of the interest the associated *Party* has in the *Occurrences*.
occurrenceChain : OccurrenceChain [1]	The *OccurrenceChains* of interest to some *Party*.

8.2.4 Occurrence

A *Occurrence* or "happening" of importance in a domain in some context.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

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

Table 14. Occurrence Attributes and/or Associations

Property/Association	Description
end : DateTime [0..1]	The *DateTime* of the end of the *Occurrence*.
kind : OccurrenceKind [0..1]	A reference to a definition of the specific kind of Occurrence.
location : Location [0..1]	The location at which an *Occurrence* took place.
occurrenceDependencies : OccurrenceDependency [0..*]	A dependency on the subject(s) of the *Occurrence*.
predecessor : OccurrenceRelationship [0..*]	A dependency on a preceding *Occurrence*.
rationale : Rationale [0..*]	The *Rationale* given for the *Occurrence*.
resultingRelationship : ElementRelationship [0..*]	The relationships generated by the *Occurrence*.
role : OccurrenceRole [*]	A role played by some *Party* in an *Occurrence*.
start : DateTime [0..1]	The *DateTime* of the start of the *Occurrence*.
subchain : OccurrenceChain [0..1]	An *OccurrenceChain* that is encapsulated by the *Occurrence* - essentially a "sub-chain".
subOccurrences : Occurrence [0..*]	A set of *Occurrences* that happen as part of the parent *Occurrence.* These *Occurrences* are normally part of a "sub-chain".
type : OccurrenceType [0..1]	The type of an *Occurrence*.

8.2.5 OccurrenceBranchNode

A kind of *OccurrenceGraphNode* that allows for branching or other kinds of connections between other *OccurrenceGraphNodes*.

Generalizations

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

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

Properties

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

8.2.6 OccurrenceChain

A succession of *Occurrences* (events or activities) that have happened in the life of some *NamedElement* that are of interest to some *Party*.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

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

Table 15. OccurrenceChain Attributes and/or Associations

Property/Association	Description
interestedParty : InterestedParty [0..*]	The *Parties* that have some interest in *Occurrences* related to the `subject` elements.
occurrenceHistory : Occurrence [0..*]	A set of *Occurrences* that comprise the chain.
occurrenceLink : OccurrenceRelationship [0..*]	The *OccurrenceRelationship(s)* that show(s) the relationship(s) between *Occurrences* in the chain.
subject : Entity [0..*]	The element(s) that is(are) the result of the *Occurrences* in the chain.
type : OccurrenceChainType [0..1]	The type of the *OccurrenceChain*.

8.2.7 OccurrenceChainType

An *OccurrenceChainType* is a kind of *ElementType* that captures a specification for a series potential *Occurrences* that are expected in a particular context. An *OccurrenceChainType* captures this specification through the occurrenceTypeGraph property - a graph of *OccurrenceGraphNodes* and *OccurrenceTransitionTypes*.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

Table 16. OccurrenceChainType Attributes and/or Associations

Property/Association	Description
interestedParty : InterestedParty [0..*]	The parties that are interested in the "lifecycle" specified by the *OccurrenceChainType.*
occurrenceType : OccurrenceType [1..*]	The `occurrenceType` derived property is based on the series of relationships between from *OccurrenceChainType* through other classes to *OccurrenceType*: `OccurrenceChainType.occurrenceTypeGraph.occurrenceNode.occurrenceType`.
occurrenceTypeGraph : OccurrenceTypeGraph [0..1]	A graph of *OccurrenceTypes* that specifies the sequencing of expected *Occurrences* in the lifecycle of an entity of interest to one or more *InterestedParties*.

subjectType : ElementType [1]	The subject of the *OccurrenceChainType*.

8.2.8 OccurrenceDependency

A type of relationship that records the dependence on an entity of interest for some particular purpose. That purpose is captured as the `role`.

OccurrenceDependencies indicate how *Entities* are used within an *Occurrence*.

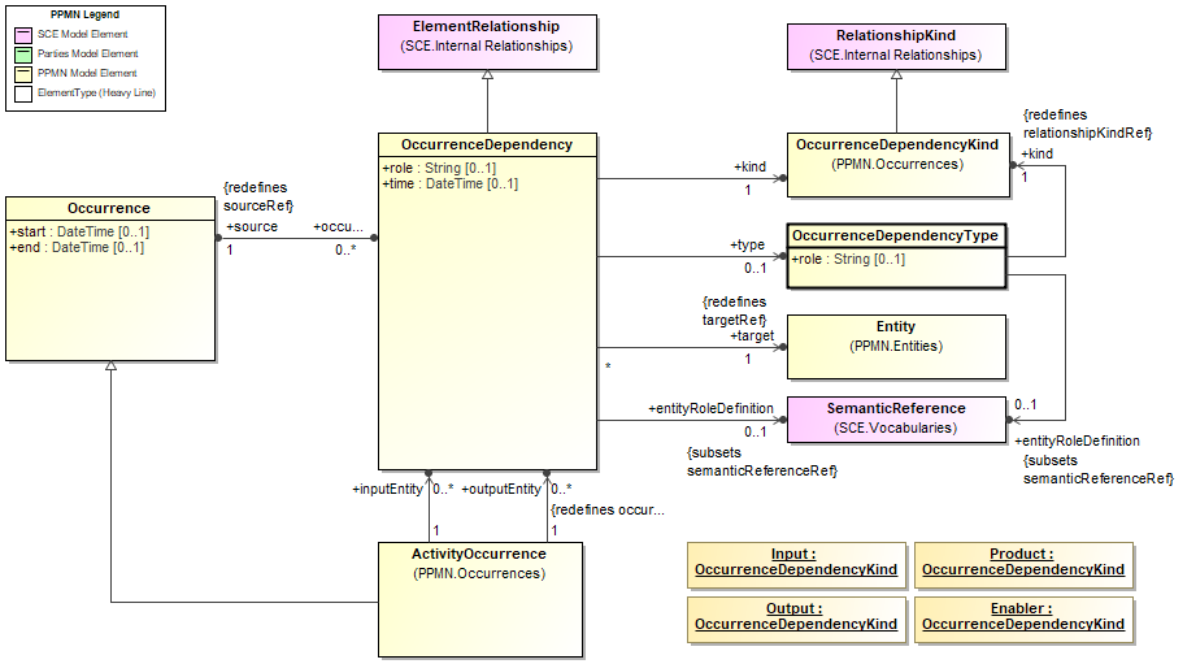


Figure 13: OccurrencesDependencies

Generalizations

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

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

Properties

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

Table 17. OccurrenceDependency Attributes and/or Associations

Property/Association	Description
entityRoleDefinition : SemanticReference [0..1]	A *SemanticReference* to a definition of the way the *Entity* was used in the *Occurrence*.
kind : OccurrenceDependencyKind [1] default: Output	A description of the type of dependency an *OccurrenceType* has on an *EntityType*. See *RelationshipKind*, below, for more details.

relationshipKind : RelationshipKind [1] default: Dependency	A description of the type of the relationship. See *RelationshipKind*, below, for more details. This property is read only and set to Dependency.
role : String [0..1]	The role of the target element in the source *Occurrence*.
source : Occurrence [1]	The *Occurrence* that has some dependency on the target *NamedElement*.
target : Entity [1]	The *NamedElement* on which some *Occurrence* depends.
time : DateTime [0..1]	The time that the *Occurrence* had the dependency on the *Entity*.
type : OccurrenceDependencyType [0..1]	The type of the *EntityDependency*.

8.2.9 OccurrenceDependencyKind

A class indicating the kind of dependency an *Occurrence* has on an *Entity.*

Generalizations

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

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

Properties

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

8.2.10 OccurrenceDependencyType

A kind of *ElementRelationship* that captures a dependency of a type of *Occurrence* on a particular type of entity and the role the entity plays in that type of *Occurrence*.

OccurrenceRoleTypes indicate how *Parties are expected to* participate in an *Occurrence*.

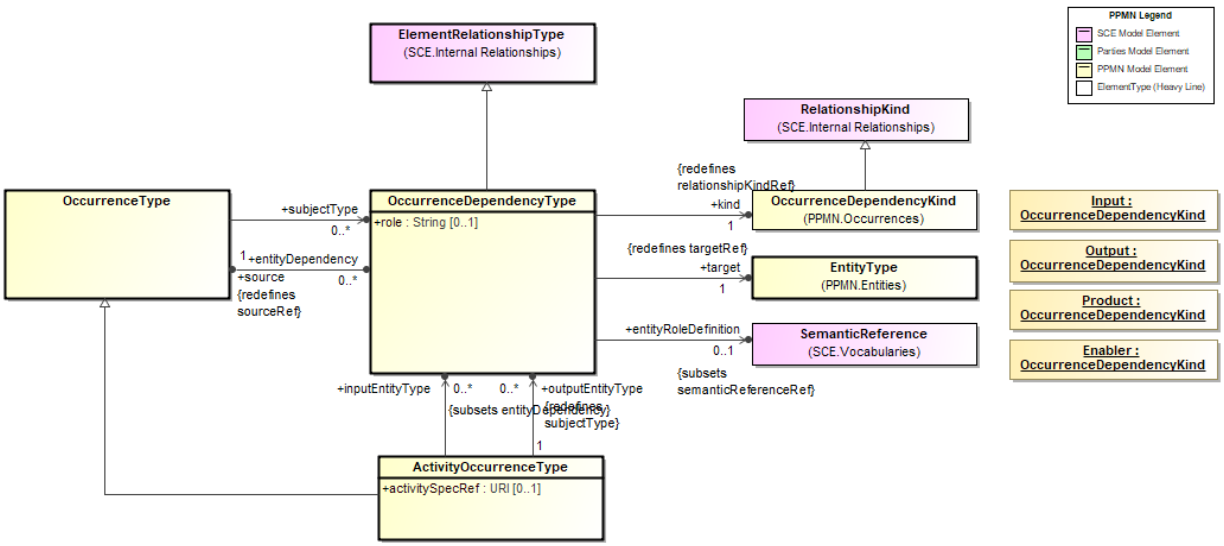


Figure 14: Occurrence Dependency Types

Generalizations

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

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

Properties

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

Table 18. OccurrenceDependencyType Attributes and/or Associations

Property/Association	Description
entityRoleDefinition : SemanticReference [0..1]	A *SemanticReference* to a definition of the way the *EntityType* is expected to be used in the *OccurrenceType*.
kind : OccurrenceDependencyKind [1] default: Output	A description of the type of dependency an *OccurrenceType* has on an *EntityType*. See EntityDependencyKind for more details.
role : String [0..1]	The role of the *ElementType* in the *OccurrenceType*.
source : OccurrenceType [1]	The *OccurrenceType* whose instances are the source of instances of the *ElementType*.
target : EntityType [1]	The *ElementType* on which the *OccurrenceType* depends.

8.2.11 OccurrenceGraphNode

A type of graph *Node* that is particular to an *OccurrenceTypeGraph*.

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

8.2.12 OccurrenceGraphTransition

A type of Link in a *OccurrenceTypeGraph* definition from one *OccurrenceType* to another.

Generalizations

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

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

Properties

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

Table 19. OccurrenceGraphTransition Attributes and/or Associations

Property/Association	Description
relationshipKind : RelationshipKind [1] default: Transition	A description of the type of the relationship. See *RelationshipKind*, below, for more details. This property is read only and set to Transition.
source : OccurrenceGraphNode [1]	The *OccurrenceGraphNode* from which the transition leaves.
target : OccurrenceGraphNode [1..*]	The *OccurrenceGraphNode* to which the transition leads.
transitionRule : Rule [0..*]	The *Rules* that constrain the *OccurrenceTransitionType*.

8.2.13 OccurrenceKind

A class indicating the specific kind of *Occurrence.*

Generalizations

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

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

Properties

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

8.2.14 OccurrenceRelationship

A kind of *ElementRelationship* that captures the fact that one *Occurrence* has a relationship to another for some reason. Examples include an *Occurrence* using an Entity created by another *Occurrence*. This usage implies that the first *Occurrence* depended on the second *Occurrence* for that *Entity*. In this way, an *OccurrenceChain* can be built by capturing and analyzing the relationships and generating the implied chain.

Generalizations

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

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

Properties

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

Table 20. OccurrenceRelationship Attributes and/or Associations

Property/Association	Description
source : Occurrence [1]	The dependent *Occurrence*.
target : Occurrence [1]	The *Occurrence* on which the source *Occurrence* depends.

8.2.15 OccurrenceRole

A role played by some *Party* in an *Occurrence*.

OccurrenceRoles indicate how a *Party* participated in an *Occurrence*.

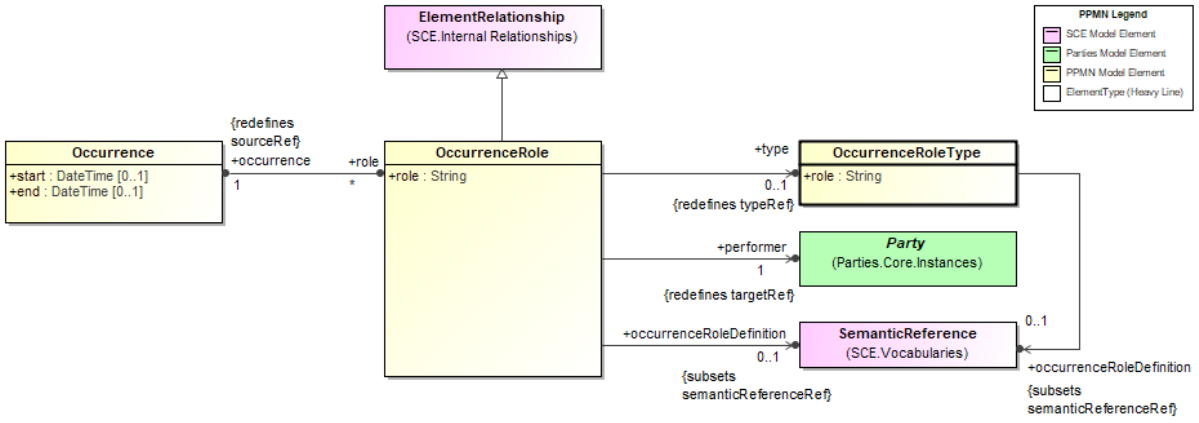


Figure 15: OccurrencesRoles

Generalizations

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

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

Properties

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

Table 21. OccurrenceRole Attributes and/or Associations

Property/Association	Description
occurrence : Occurrence [1]	The *Occurrence* in which the *Party* plays the role.
occurrenceRoleDefinition : SemanticReference [0..1]	A *SemanticReference* to a definition of the role the *Party* played in the *Occurrence*.
performer : Party [1]	The *Party* that plays the role in an *Occurrence* specified by the *OccurrenceRole*.
role : String []	A textual description of the actual role played by the `performer` in the activity.
type : OccurrenceRoleType [0..1]	The type of the role played by the `performer` *Party* in the *Occurrence*.

8.2.16 OccurrenceRoleType

A specification of the type of party expected to play a role an *OccurrenceType*.

OccurrenceTypes support the definition of expected Occurrences in a Pedigree or Provenance Chain. Essentially, *OccurrenceTypes* represent *Occurrence* instances that are expected to with respect to entities of a particular type from the perspective of the *InterestedParties.* These expected *OccurrenceTypes* can be organized into graphs, *OccurrenceTypeGraphs*, that show an expected sequence or "chain" of those types of *Occurrences*. Further, *OccurrenceTypes* can optionally have sub-chain types so that *OccurrenceTypeGraphs* can be nested within one another. *OccurrenceTypeRole* captures roles expected to be played by *Parties*.

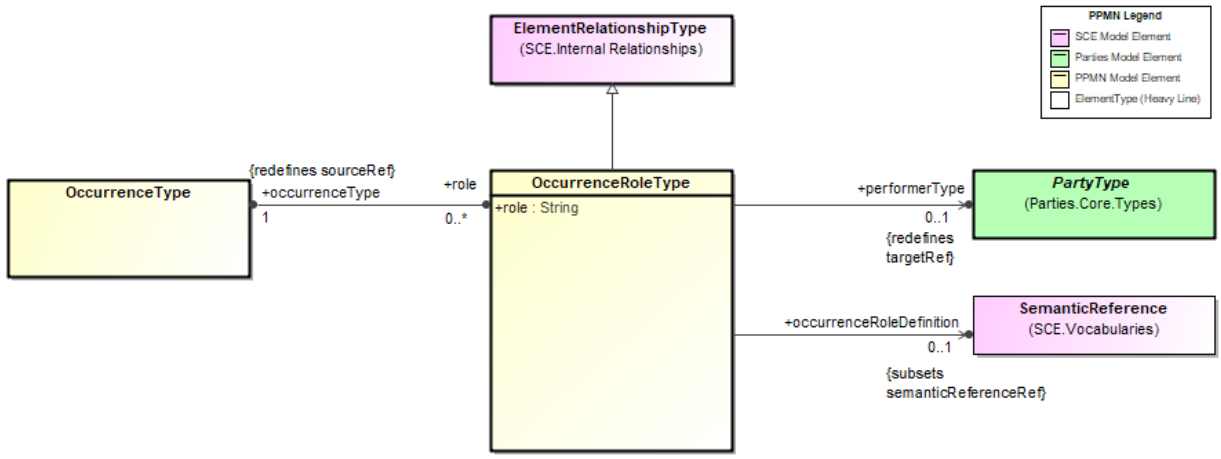


Figure 16: Occurrence Role Types

Generalizations

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

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

Properties

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

Table 22. OccurrenceRoleType Attributes and/or Associations

Property/Association	Description
occurrenceRoleDefinition : SemanticReference [0..1]	A *SemanticReference* to a definition of the role the *PartyType* is expected to play in the *OccurrenceType*.
occurrenceType : OccurrenceType [1]	The type of *Occurrence* in which the `expectedPerformer` to perform in the role.
performerType : PartyType [0..1]	The *Party* that is expected to perform in a particular role in an *Occurrence*.
role : String []	A textual description of the role in the *Occurrence*.

8.2.17 OccurrenceType

The type or specification of an *Occurrence* that may happen or be of interest. An *OccurrenceType* may have a `subChainType` enabling nesting of *OccurrenceChainTypes*.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

Table 23. OccurrenceType Attributes and/or Associations

Property/Association	Description
entityDependency : OccurrenceDependencyType [0..*]	A dependency on the *ElementTypes* that are involved in this *OccurrenceType*.
kind : OccurrenceKind [0..1]	A reference to a definition of the specific kind of Occurrence.
location : Location [0..*]	The location at which *Occurrences* of type *OccurrenceType* are planned or expected to happen.
locationTypes : LocationType [0..*]	The types of *Locations* at which *Occurrences* of type *OccurrenceType* are planned or expected to happen.
rationale : Rationale [0..1]	The *Rationale* given for the *OccurrenceType*.
resultingRelationshipType : ElementRelationshipType [0..*]	The *ElementRelationshipTypes* that exist as a result of *Occurrences* of type *OccurrenceType*.
role : OccurrenceRoleType [0..*]	A set of *OccurrenceTypeRoles* that specify the role a *Party* is expected to play in an *Occurrence*.
subChain : OccurrenceChainType [0..1]	An *OccurrenceChainType* that is encapsulated within the *OccurrenceType* to create a "subchain".
subjectType : OccurrenceDependencyType [0..*]	A dependency on the *ElementTypes* that are the subject of this OccurrenceType.

8.2.18 OccurrenceTypeGraph

A type of Graph that captures the OccurrenceTypes that are expected in the lifecycle of one or more EntityTypes.

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 24. OccurrenceTypeGraph Attributes and/or Associations

Property/Association	Description
occurrenceNode : OccurrenceGraphNode [1..*]	The *OccurrenceGraphNodes* included in the *OccurrenceTypeGraph*.
transition : OccurrenceGraphTransition [0..*]	The *OccurrenceTypeTransitions* included in the *OccurrenceTypeGraph*.

8.2.19 OccurrenceTypeUsage Node

A kind of *OccurrenceGraphNode* that identifies the usage of an *OccurrenceType* in an *OccurrenceTypeGraph*.

Generalizations

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

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

Properties

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

Table 25. OccurrenceTypeUsage Node Attributes and/or Associations

Property/Association	Description
occurrenceType : OccurrenceType [1]	The *OccurrenceType* that the node represents.

8.2.20 PPMNRelationshipKind

A class indicating the kind of relationship between two **PPMN** elements.

Generalizations

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

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

Properties

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

8.2.21 Rule

A condition that can be evaluated in some context as being either True or False.

Generalizations

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

Properties

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

8.3 Pedigree

The *Pedigree* package contains elements necessary to capture the lineage or pedigree of *Entities* along with the *Occurrences* that resulted in that lineage.

8.3.1 Pedigree Occurrences

The *Pedigree Occurrences* package contains elements necessary to capture the events or activities, i.e. the *Occurrences*, that affect the lifecycle of *Entities*.

PedigreeChains record the actual events or processes that happen as part of the history of an entity of interest. *PedigreeChains* also record a reference to the entity to which the *Occurrences* relate through the `entity` property. Conceptually, *PedigreeChains* are "instances" of *PedigreeChainTypes* and as such may be governed by the relations established in the *PedigreeChainType*. These occurrences represent actual events or activities in the history of one or more *Entities* that are of interest to some *Party*.

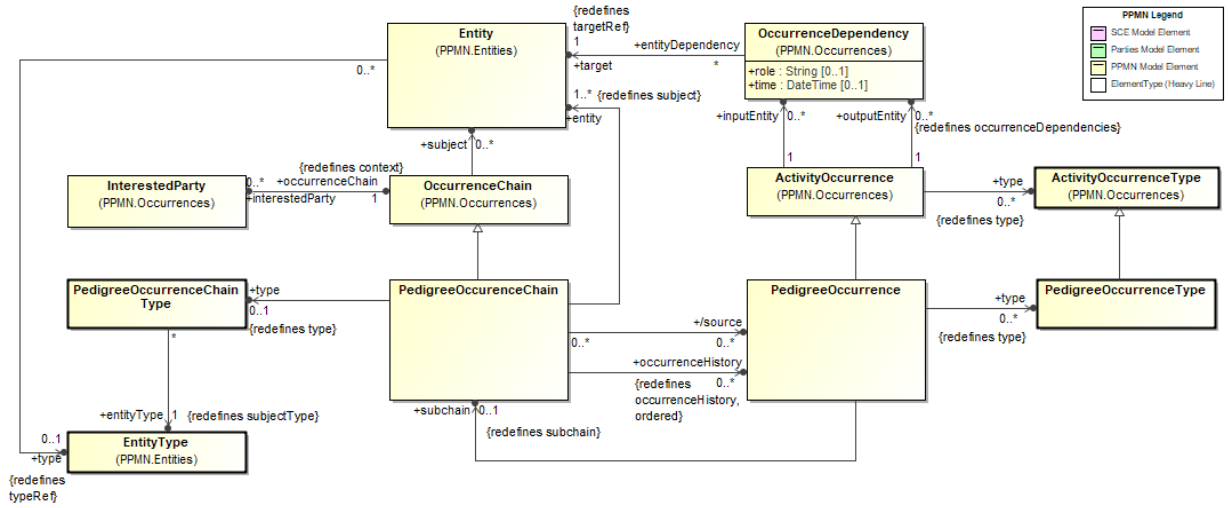


Figure 17: Pedigree Occurrence Chains - Overview

PedigreeOccurrence is a kind of *ActivityOccurrence* that affects the lifecycle of one or more *Entities*. PedigreeOccurrences take

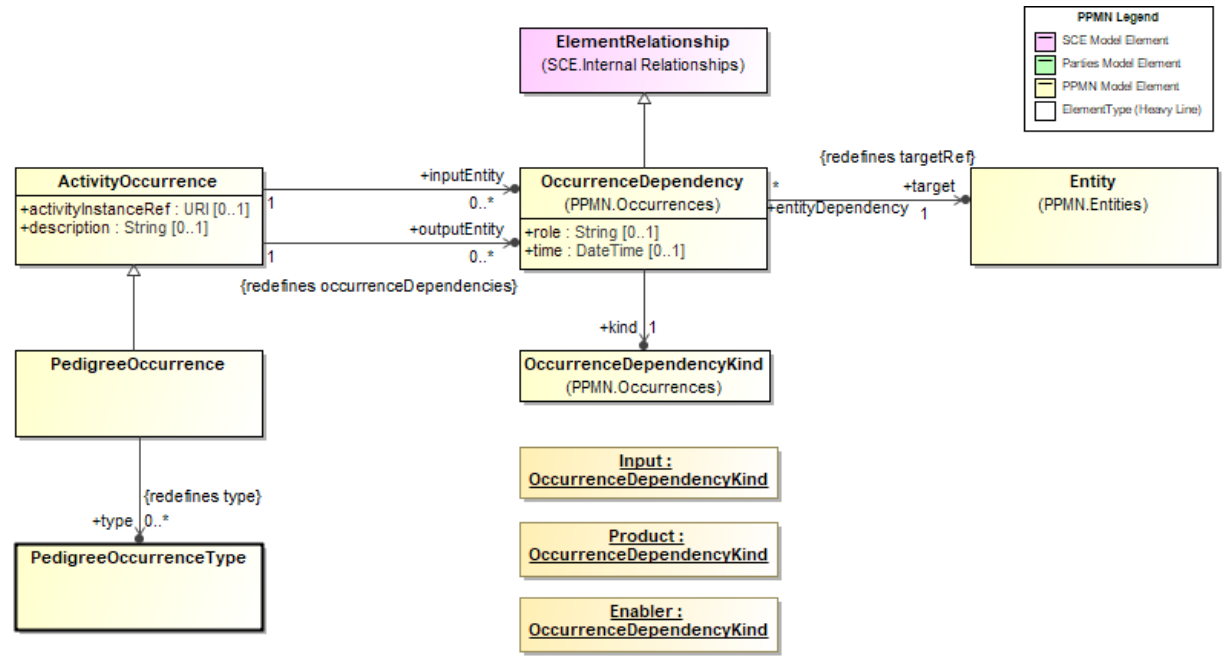


Figure 18: Pedigree Occurrences

PedigreeOccurrenceChains record the actual *PedigreeOccurrences* that happen as part of the `occurrenceHistory` property, an ordered list. *PedigreeOccurrenceChains* include a reference to the *Entity* or *Entities* to which the *Occurrences* relate through the `entity` property. *PedigreeOccurrenceChains* are essentially instances of *PedigreeChainTypes* and as such are governed by the relations established in the *PedigreeChainType*.

PedigreeOccurrences are instances of *PedigreeOccurrenceTypes*. These occurrences represent actual events or activities in the history of an *Entity* that is of interest to some *Party*.

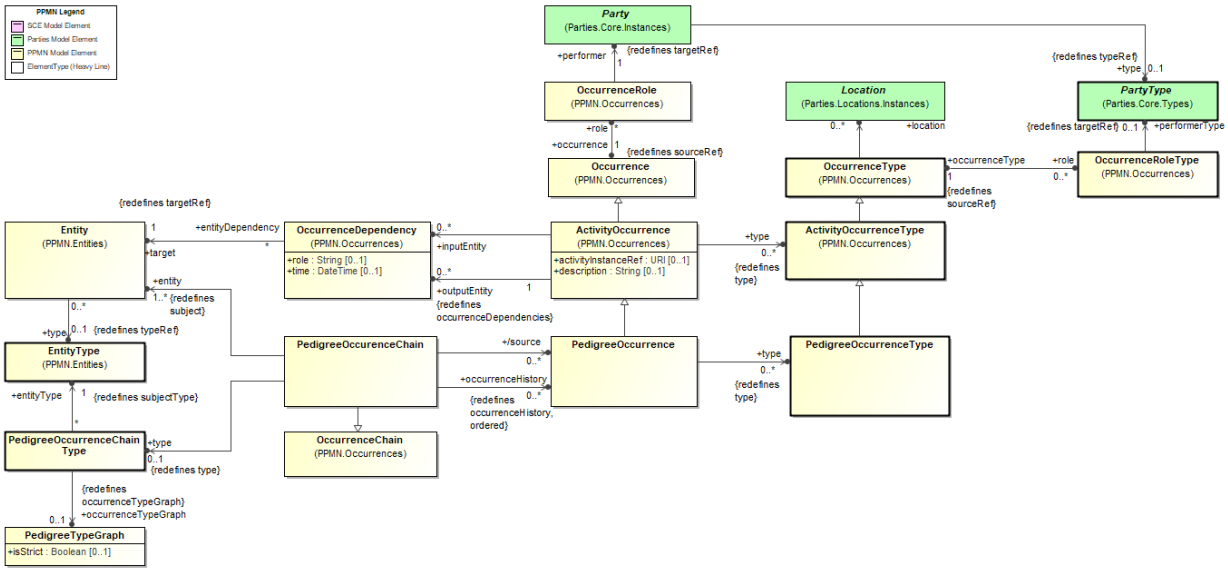


Figure 19: Pedigree Occurrence Chains

PedigreeChainType supports the definition of types of occurrences expected in *PedigreeChains* related to an *EntityType* in which some *Party* is interested. *PedigreeChainTypes* are modeled as simple graphs so that rich definitions of entity lifecycles can be created (though they are not required). The model also supports simple definitions of valid *PedigreeOccurrenceTypes* or no lifecycle definitions at all.

InterestedParty is a kind of *PartyRole* that indicates that a *Party* has some interest in the with respect to an entity. *PedigreeChainTypes* are specific to one or more *InterestedParties*. As an example, an automobile manufacturer may be interested a set of occurrences related to the building of a car such as StartAssembly, InstallEngine, PaintCar, TestCar, and ShipCar. A dealership on the other hand would likely be interested in tracking other events such as BuildCar, ShipCar, ReceiveCar, and SellCar.

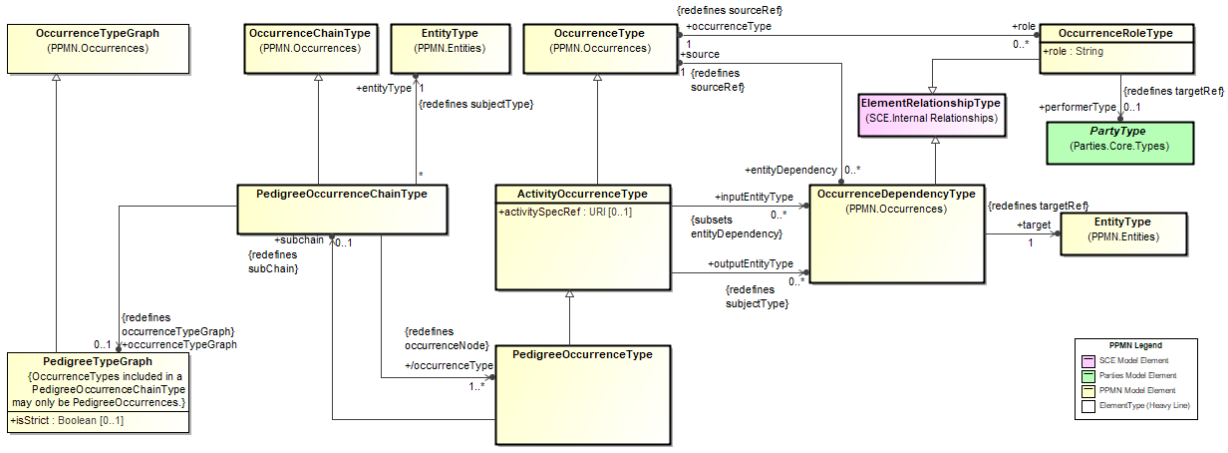


Figure 20: Pedigree Occurrence Chain Type

PedigreeChainType supports the definition of types of occurrences expected in *PedigreeChains* related to an entity type in which some *Party* is interested. *PedigreeChainTypes* are modeled as simple graphs so that rich definitions of entity lifecycles can be created (though they are not required). The model also supports simple definitions of

valid *PedigreeOccurrenceTypes* or no lifecycle definitions at all.

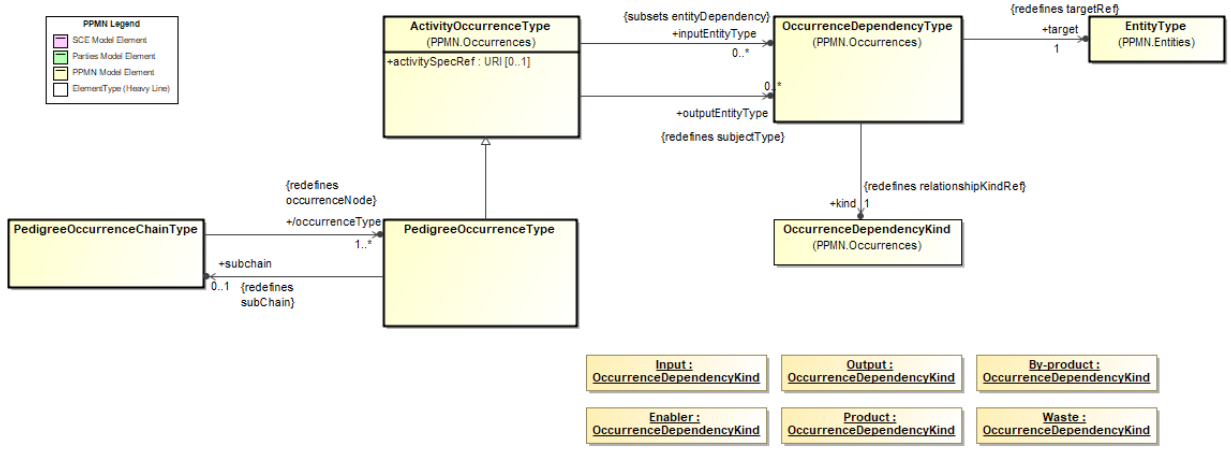


Figure 21: Pedigree Occurrence Types

The lineage of an *Entity*, herein referred to as its "pedigree" or "pedigree chain", is a lattice comprising *Entities* as nodes and derivations (*DerivedFrom* relationships) as edges. Pedigree chains are created by *Occurrences* that result in some number of *Entities* being used to create one or more new *Entities* or evolve one or more existing *Entities*. These *Occurrences* result one or more derivations between "input" *Entities* and the "output" *Entities*.

Given that a particular *Occurrence* may encapsulate a sub-chain of *Occurrences*, derivations may involve a series of one or more *Occurrences* that create or evolve an entity of interest into another. In these cases, the *Occurrences* that comprise the sub-chain would also potentially result in derivations that would combine to result in the derivations of the containing *Occurrence*. As stated above, derivations are noted in the form of a *DerivedFrom* relationship between one *Entity* that is the `derivee` and another that is the `derivedEntity`. The derivation may be related to an *ActivityOccurrence* that caused the transformation. This may specifically be a *PedigreeOccurrence* but may also be a more general *ActivityOccurrence*. Often, the activities that result in derivations are not easily tracked or quantified and so just noting the *Entity* or *Entities* from which the entity of interest is derived is all that is necessary or in some cases even possible.

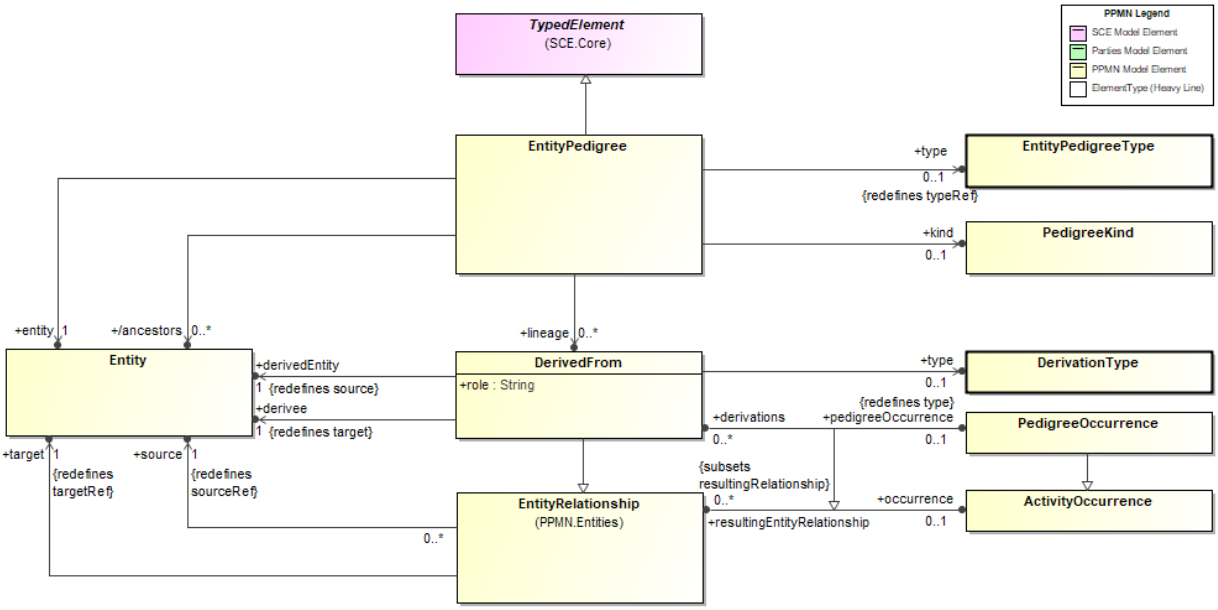


Figure 22: Pedigree "Chains"

EntityPedigreeTypes support the ability to define different kinds of pedigree or lineage of particular kinds of *Entities*. This is accomplished by specifying the *EntityTypes* and the types of derivations between them. Derivations involve a series of one or more *Occurrences* that create or evolve an entity of interest into another. Derivations are noted in the form of a *DerivedFrom* relationship between one *Entity* that is the `derivee` and another that is the `derivedEntity`. To specify the expected type of derivation between two *Entities* **PPMN** provides the *DerivationType* element. In addition, **PPMN** specifies three types of derivation: revision, quotation, and sourcing. (See section 8.3.2, below, for further explanation.)

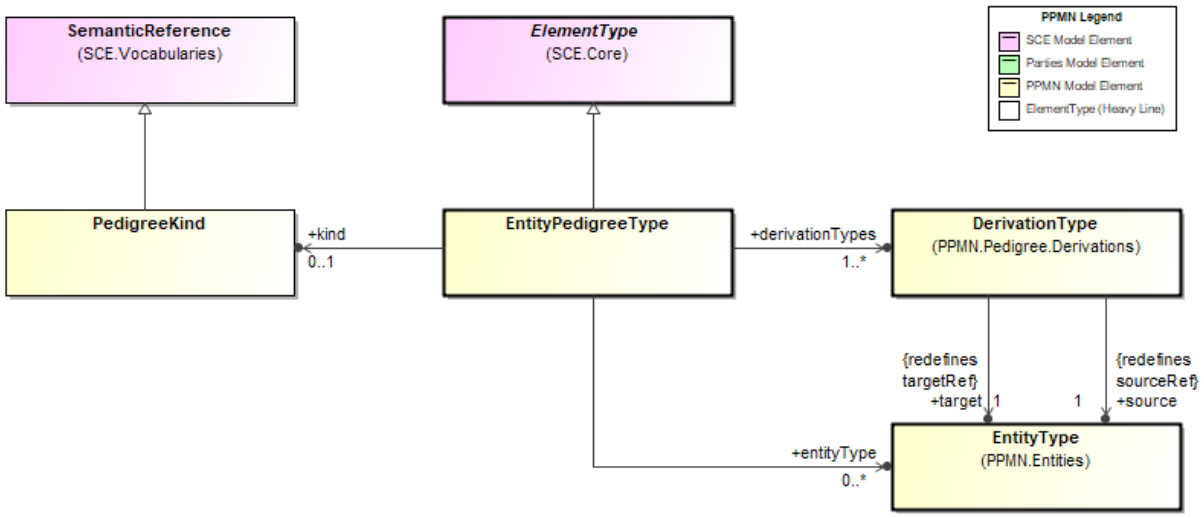


Figure 23: Pedigree Chains Types

8.3.1.1 EntityPedigree

The class representing the pedigree or lineage of an *Entity*.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

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

Table 26. EntityPedigree Attributes and/or Associations

Property/Association	Description
ancestors : Entity [0..*]	The set of *Entities* from which the `entity` was derived. This is a derived property determined by walking the set of *DerivedFrom* relationships from *Entity* to *Entity* until the end of each path of the directed acyclic graph (DAG).
entity : Entity [1]	The *Entity* to which the pedigree applies.
kind : PedigreeKind [0..1]	A specification of the kind of pedigree the EntityPedigree captures.

lineage : DerivedFrom [0..*]	The set of *DerivedFrom* relationships that led to the creation and/or evolution of the `entity`. The combination of the *DerivedFrom* relationships and the *Entities* at their ends must form a directed acyclic graph (DAG) starting with the `entity` and ending with *Entities* that were created by some *Occurrence* or whose origin is unknown.
type : EntityPedigreeType [0..1]	

8.3.1.2 EntityPedigreeType

The type of pedigree or lineage between *Entities* of type entityType.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

Table 27. EntityPedigreeType Attributes and/or Associations

Property/Association	Description
derivationTypes : DerivationType [1..*]	The types of derivations that are captured by the EntityPedigreeType.
entityType : EntityType [0..*]	The *EntityType(s)* to which the EntityPedigreeType applies.
kind : PedigreeKind [0..1]	The kind of entity pedigree or lineage the EntityPedigreeType represents.

8.3.1.3 PedigreeKind

A class that indicates the kind of pedigree or lineage between *Entities*.

Generalizations

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

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

Properties

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

8.3.1.4 PedigreeOccurenceChain

A succession of *PedigreeOccurrences* that have happened in the life of an entity that is of interest to some *Party*.

Generalizations

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

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

Properties

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

Table 28. PedigreeOccurenceChain Attributes and/or Associations

Property/Association	Description
entity : Entity [1]	The *Entity* or *Entities* for which the *PedigreeChain* represents the history of *PedigreeOccurrences*.
occurrenceHistory : PedigreeOccurrence [0..*]	A sequence of *PedigreeOccurrences* that represent the history of *PedigreeOccurrences* that took place with respect to a particular entity.
source : PedigreeOccurrence [0..*]	The *PedigreeOccurrences* that were the original sources for ancestor entities of the subject entity.
type : PedigreeOccurrenceChainType [0..1]	The type of the *PedigreeChain*.

8.3.1.5 PedigreeOccurrence

An *ActivityOccurrence* in the lifecycle of an entity related to the source or evolution of that entity that is of interest to some *Party*.

Generalizations

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

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

Properties

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

Table 29. PedigreeOccurrence Attributes and/or Associations

Property/Association	Description
derivations : DerivedFrom [0..*]	Derivations created as a result of the PedigreeOccurrence.
subchain : PedigreeOccurenceChain [0..1]	A sequence of *PedigreeOccurrences* that take the `inputEntities` of the *PedigreeOccurrence* and transform them into the `outputEntities` of the *PedigreeOccurrence* and are encapsulated by the *PedigreeOccurrence*.
type : PedigreeOccurrenceType [0..*]	The type of the *PedigreeOccurrence*.

8.3.1.6 PedigreeOccurrenceChainType

A kind of *OccurrenceChainType* that captures the expected *OccurrenceTypes*, *PedigreeOccurrenceTypes*, that result in the creation or evolution of particular types of entities.

Generalizations

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

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

Properties

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

Table 30. PedigreeOccurrenceChainType Attributes and/or Associations

Property/Association	Description
entityType : EntityType [1]	The type of entity expected as a result of the chain.
occurrenceType : PedigreeOccurrenceType [1..*]	The `occurrenceType` derived property is based on the series of relationships between from *PedigreeChainType* through other classes to *PedigreeOccurrenceType*: `OccurrenceChainType.occurrenceTypeGraph.occurrenceNode.occurrenceType`.
occurrenceTypeGraph : PedigreeTypeGraph [0..1]	A graph of *PedigreeOccurrenceTypes* that are expected in the lifecycle of a particular type of entity.

8.3.1.7 PedigreeOccurrenceType

An expected type of *PedigreeOccurrence* in the lifecycle of an entity that is of interest to some *Party*.

Generalizations

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

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

Properties

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

Table 31. PedigreeOccurrenceType Attributes and/or Associations

Property/Association	Description
subchain : PedigreeOccurrenceChainType [0..1]	A *PedigreeChainType* that is encapsulated within the *PedigreeOccurrenceType* to create a "subchain".

8.3.1.8 PedigreeTypeGraph

A *PedigreeChainType* is a specification for the types of *Occurrences* that happen with respect to an entity that are of interest to a particular *Party*. If the property `isStrict`=True, then only the *Occurrences* of type *PedigreeOccurrenceType* will be included in related *PedigreeChains*. If the property is False then *Occurrences* of other types may be included in related *PedigreeChains*.

Generalizations

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

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

Properties

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

Table 32. PedigreeTypeGraph Attributes and/or Associations

Property/Association	Description
isStrict : Boolean [0..1]	A boolean that specifies whether or not adherence to the *PedigreeTypeGraph* is strict or not. If the value is True, then only the *Occurrences* of type *PedigreeOccurrenceType* will be included in related *PedigreeChains*. If the value is False then *Occurrences* of other types may be included in related *PedigreeChains*.

8.3.2 Derivations

The Derivations package contains elements that capture the derivation relationships between Entities. These elements, in conjuction with Entities, capture the lineage or pedigree of Entities.

Derivations capture the lineal relationships between Entities or Entity Snapshots. Derivations are noted in the form of a *DerivedFrom* relationship, or one of its specializations, between one *Entity* that is the `derivee` and another that is the `derivedEntity`. A derivation may be the result of a general *ActivityOccurrence* or specifically a *PedigreeOccurrence*. Please note that the activities that result in derivations are not always easily tracked or quantified and so just noting the entity from which the entity of interest is derived is all that is possible.

PPMN specifies four types of derivation: revision, quotation, sourcing, and descendant. Revision is captured in the form of the *RevisionOf* relationship. *RevisionOf* is a specialization of *DerivedFrom* and is used in situations where one entity is a revision of another as in a report or publication. Quotation is captured by the *QuotedFrom* specialization of *DerivedFrom* and specifies that part of all of one entity is a repeat of part or all of another entity, presumably some textual report or publication. The quotation may or may not be by the original author of the quoted entity. *SourcedFrom* is a specialization of *DerivedFrom* that identifies that the entity of interest came from another entity which was in turn produced by some party potentially with some special experience or knowledge. Finally, *DescendantOf* specializes *DerivedFrom* and indicates that the entity of interest is a descendant of the `ancestor` *Entity*.

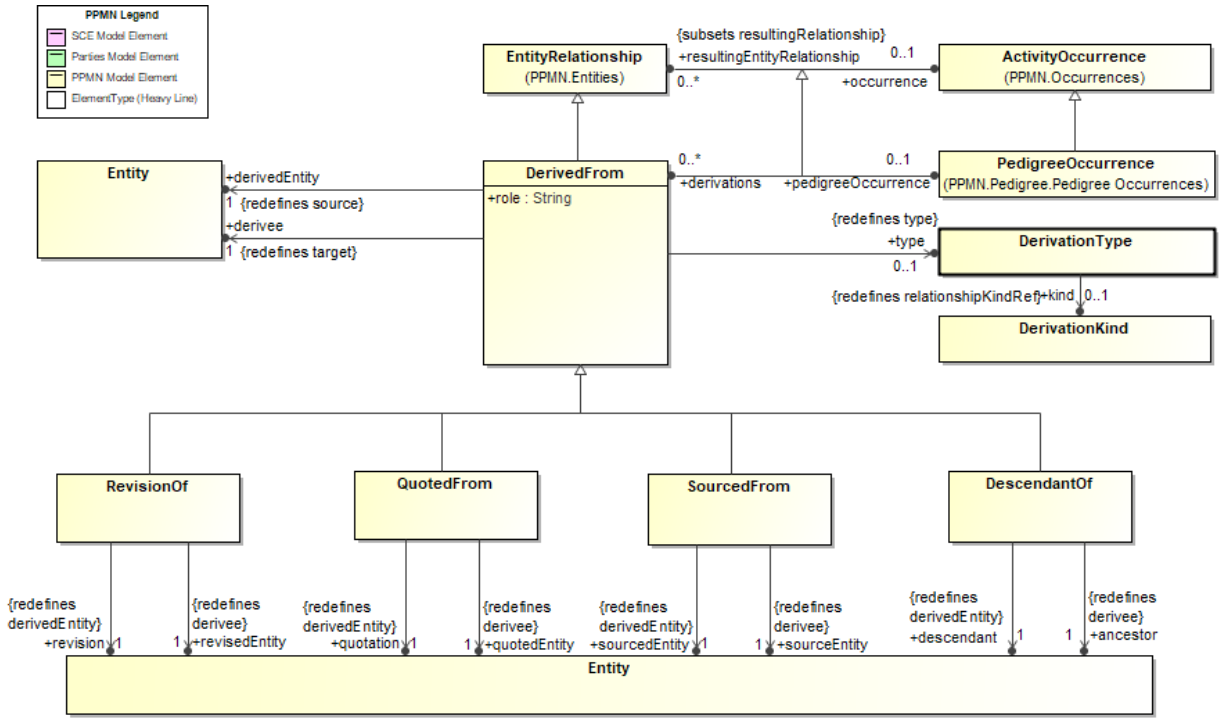


Figure 24: Derivations

DerivationTypes support the definition of the expected kinds of derivations that might result in the generation of one *EntityType* from or more others.

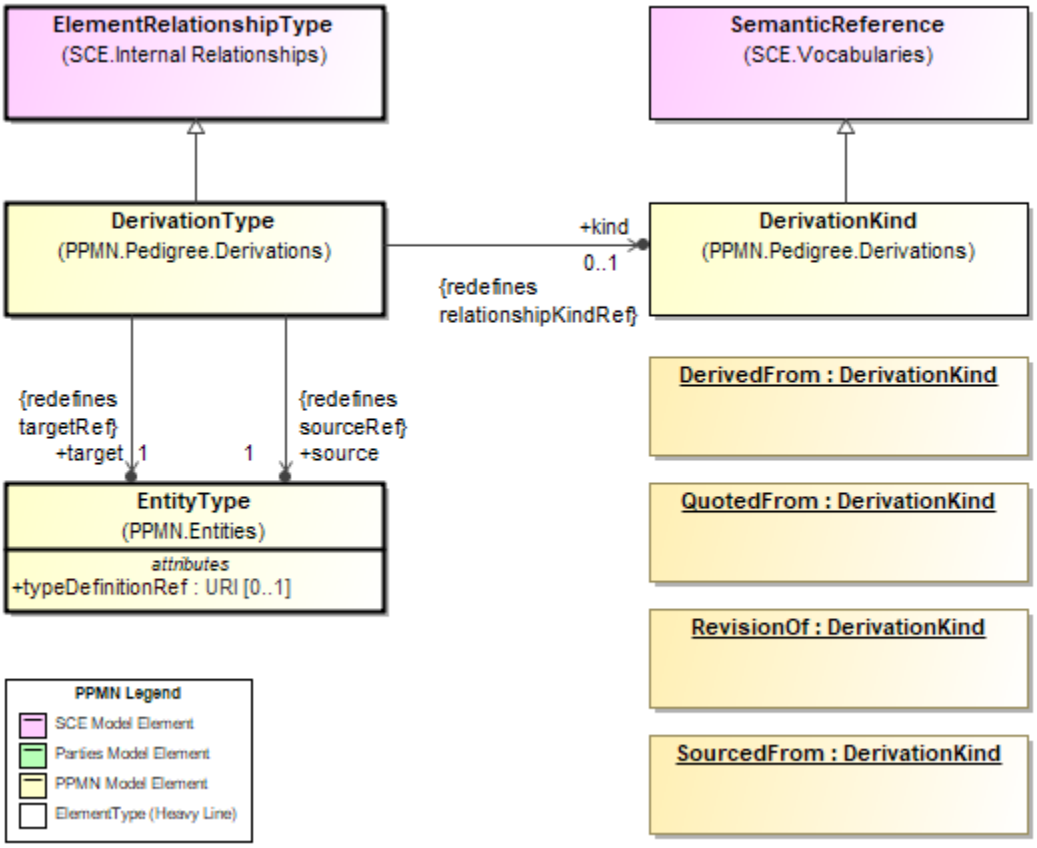


Figure 25: Derivation Types

8.3.2.1 DerivationKind

A class indicating the kind of derivation that exists between two *Entities.*

Generalizations

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

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

Properties

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

8.3.2.2 DerivationType

A kind of *ElementRelationship* that captures the type of derivation between one particular *EntityType* and another.

Generalizations

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

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

Properties

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

Table 33. DerivationType Attributes and/or Associations

Property/Association	Description
kind : DerivationKind [0..1]	A description of the kind of derivation that produced one *Entity* from another. See DeivationKind for more details.
source : EntityType [1]	The *EntityType* that was derived.
target : EntityType [1]	The *EntityType* from which the `source` *EntityType* was derived.

8.3.2.3 DerivedFrom

Derivations are noted in the form of a *DerivedFrom* relationship between one *Entity* that is the `derivee` and another that is the `derivedEntity`. The derivation may be related to an *ActivityOccurrence* that specifies the particular *Occurrence* that caused the transformation. Often, the activities that result in derivations are not easily tracked or quantified and so just noting the entity from which the entity of interest is derived is all that is necessary.

Generalizations

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

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

Properties

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

Table 34. DerivedFrom Attributes and/or Associations

Property/Association	Description
derivedEntity : Entity [1]	The *Entity* that was derived.
derivee : Entity [1]	The *Entity* from which the `derivedEntity` was derived.
pedigreeOccurrence : PedigreeOccurrence [0..1]	The *PedigreeOccurrence* that resulted in the derivation.
role : String []	A string that captures the role in the `derivationOccurrence` that produced the element.
type : DerivationType [0..1]	The type of derivation.

8.3.2.4 DescendantOf

DescendantOf is a specialization of *DerivedFrom* that identifies that the entity of interest is a descendant of another *Entity*.

Generalizations

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

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

Properties

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

Table 35. DescendantOf Attributes and/or Associations

Property/Association	Description
ancestor : Entity [1]	The ancestor *Entity*.
descendant : Entity [1]	The descendant *Entity*.

8.3.2.5 QuotedFrom

Quotation is captured by the *QuotedFrom* specialization of *DerivedFrom* and specifies that part of all of one entity is a repeat of part or all of another entity, presumably some textual report or publication. The quotation may or may not be by the original author of the quoted entity.

Generalizations

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

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

Properties

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

Table 36. QuotedFrom Attributes and/or Associations

Property/Association	Description
quotation : Entity [1]	The element that is the quotation.
quotedEntity : Entity [1]	The quoted element.

8.3.2.6 RevisionOf

Revision is captured in the form of the *RevisionOf* relationship. *RevisionOf* is a specialization of *DerivedFrom* and is used in situations where one entity is a revision of another as in a report or publication.

Generalizations

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

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

Properties

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

Table 37. RevisionOf Attributes and/or Associations

Property/Association	Description
revisedEntity : Entity [1]	The revised element.
revision : Entity [1]	The result of the revision.

8.3.2.7 SourcedFrom

SourcedFrom is a specialization of *DerivedFrom* that identifies that the entity of interest came from another entity which was in turn produced by some party potentially with some special experience or knowledge.

Generalizations

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

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

Properties

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

Table 38. SourcedFrom Attributes and/or Associations

Property/Association	Description
sourcedEntity : Entity [1]	The sourced element.
sourceEntity : Entity [1]	The entity from which the `sourcedEntity` was sourced.

8.4 Provenance

The Provenance package contains elements related to the notion of the ownership and custody of entities of interest. This includes the *Occurrences* that result in changes in the ownership or custody of those entities of interest.

ProvenanceOccurrences are specializations of *Occurrence* related to changes in ownershp or custody of an entity. *ProvenanceOccurrences are* instances of *ProvenanceOccurrenceType* or one of its specializations. Similar to *OccurrenceType*, *ProvenanceOccurrenceType* is a specification of "expected" *ProvenanceOccurrences.* They capture the *Parties* expected to be involved in the instances. Expected types of entities to which the occurrences refer are noted through the `entityType` property.

A *ProvenanceChain* records the provenance-related events that happen as part of the lifecycle of an entity. These events are recorded as part of the `occurrenceHistory` property, an ordered list of *ProvenanceOccurrences*. A *ProvenanceChain* also records a reference to the entity to which the *Occurrences* relate through the `entity` property. *ProvenanceChains* are essentially instances of Provenance*ChainTypes* and as such are governed by the relations established in the *ProvenanceChainType*. If the *ProvenanceChainType* `isStrict` property is set to "True" then the types of occurrences maintained in the *ProvenanceChain* are constrained to those included in the *ProvenanceChainType*.

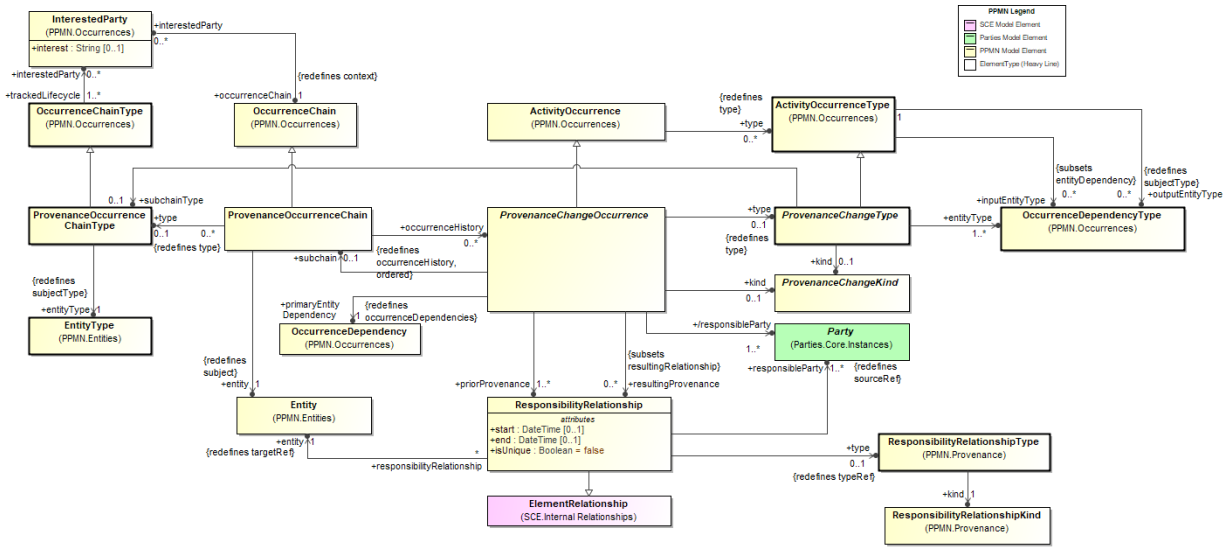


Figure 26: Provenance Occurrence Chains

ProvenanceChains, *ProvenanceChainTypes*, *ProvenanceOccurrences*, and *ProvenanceOccurrenceTypes* follow the same pattern that **PPMN** establishes for *Occurrences*. This pattern supports the "nesting" of *ProvenanceChains* within *ProvenanceOccurrences*. This pattern allows for encapsulation of parts of a chain where the details of the *ProvenanceOccurrences* of that part of a larger chain are either not known initially or are not deemed important in some context.

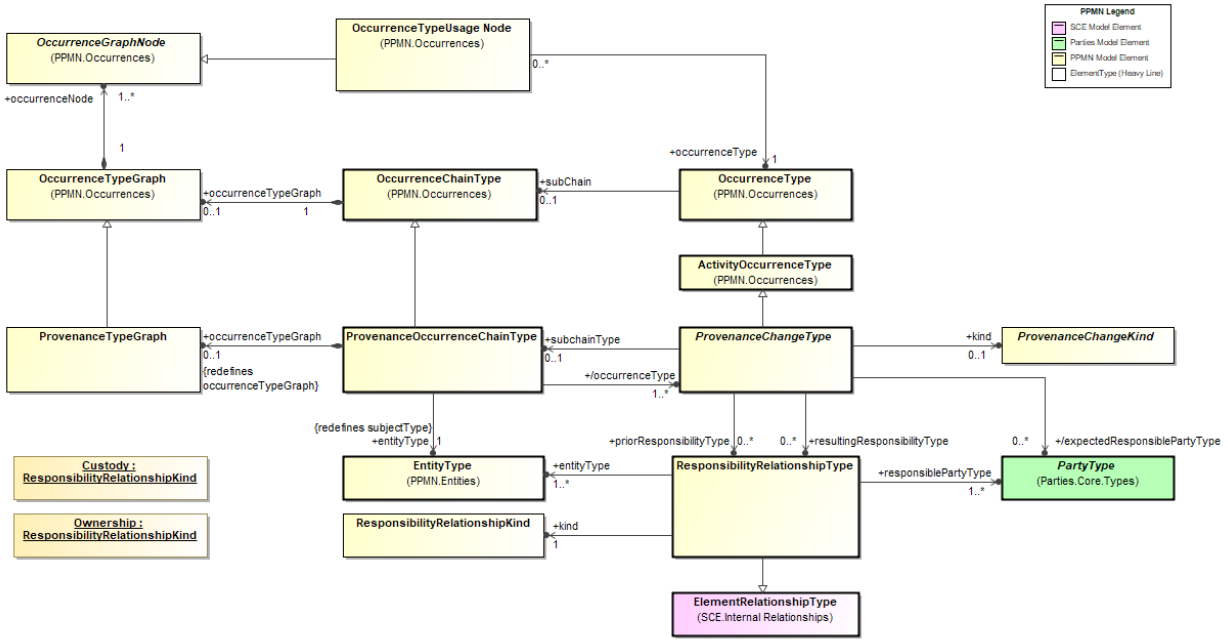


Figure 27: Provenance Occurrence Chain Types

In addition to tracking changes in ownership or custody for an entity of interest over time, stakeholders also require the ability to make direct statements about who owns or has custody of an entity at a particular point in time. The *Ownership* and *Custody* classes provide this capability. Both *Ownership* and *Custody* specializations of *ResponsibilityRelationship* and, as such, capture the *Party* that owns or has custody of, respectively, a particular *Entity* for a particular period of time. These provenance "records" can either be maintained in real time or generated

based on *Occurrences* that have been tracked for an entity.

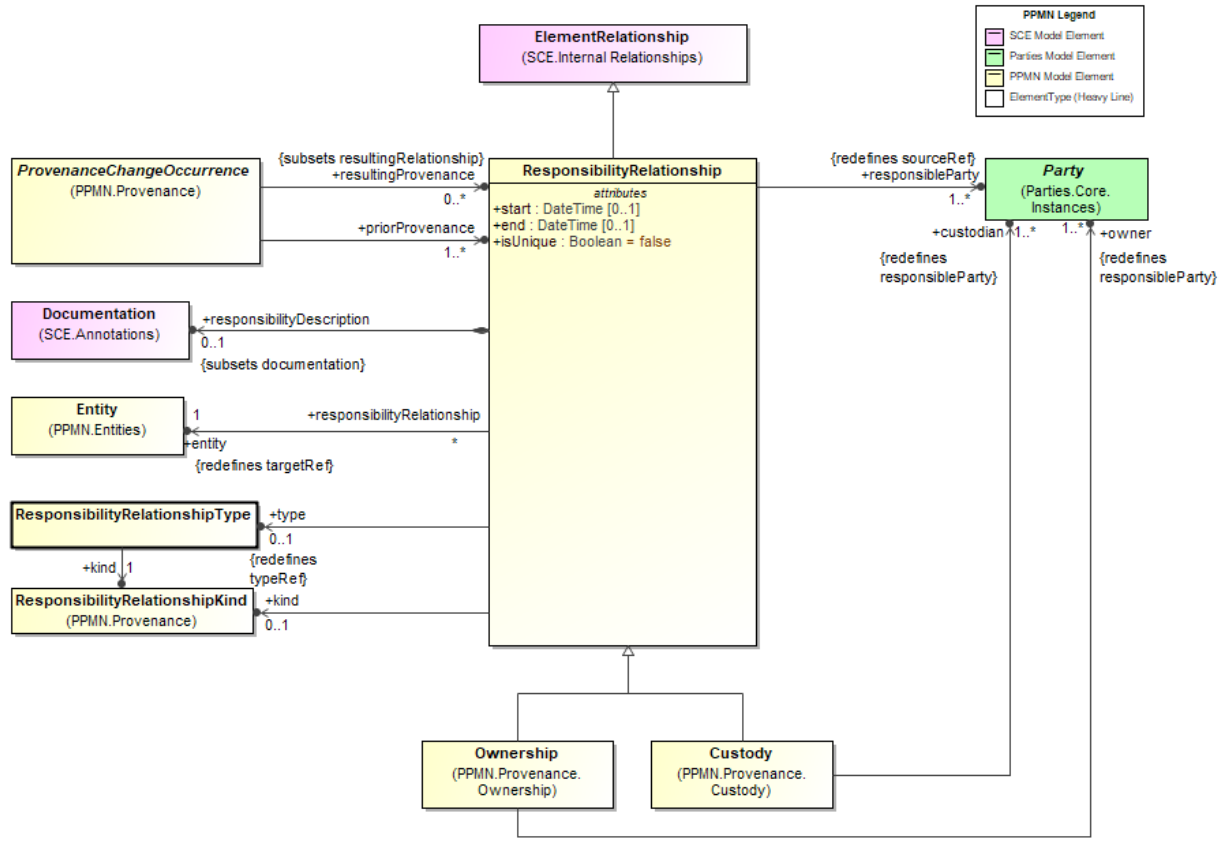


Figure 28: Provenance "Records"

TBD.

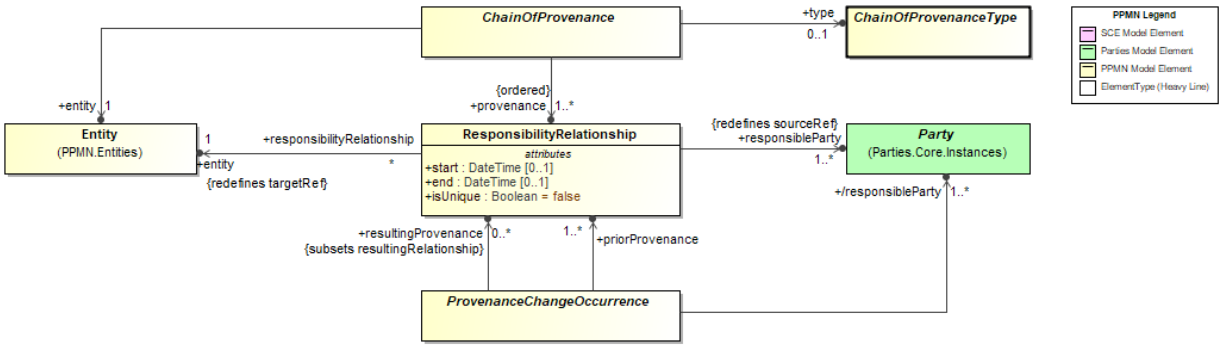


Figure 29: Chain of Provenance

TBD.

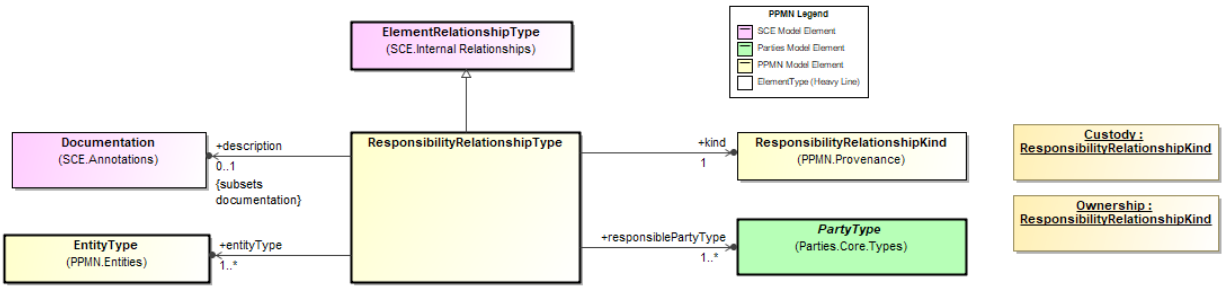


Figure 30: Provenance Record Types

TBD.

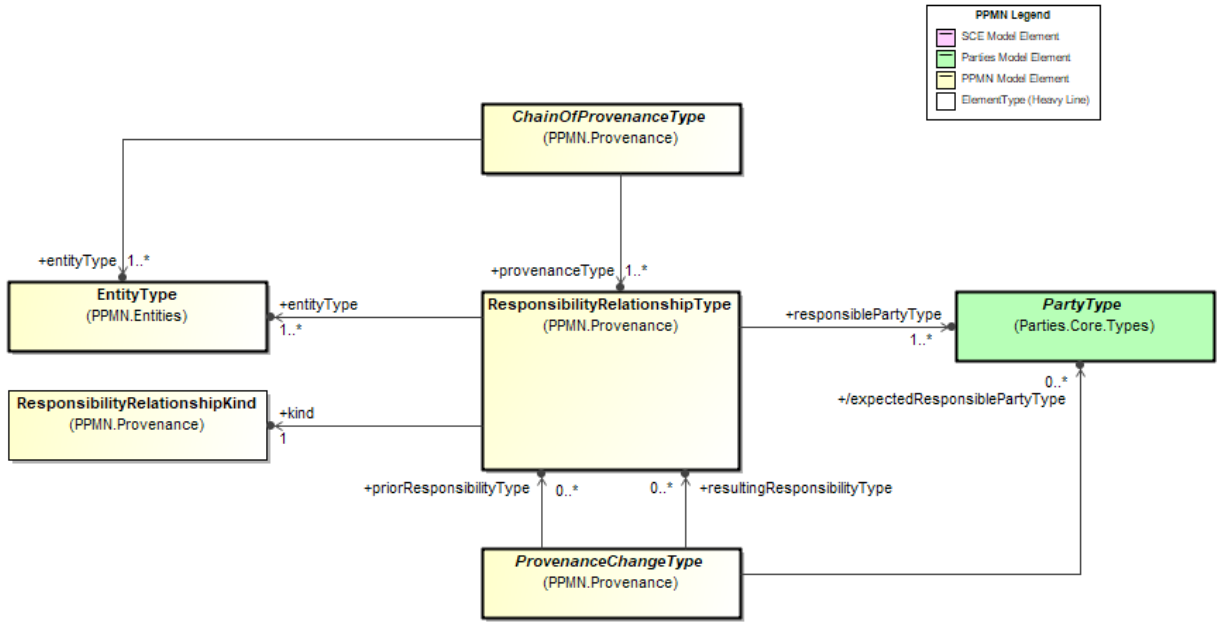


Figure 31: Chain of Provenance Types

8.4.1 ChainOfProvenance

An ordered set of *ResponsibilityRelationships* that captures the provenance of a particular entity over the course of its lifecycle.

Generalizations

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

Properties

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

Table 39. ChainOfProvenance Attributes and/or Associations

Property/Association	Description
entity : Entity [1]	The entity to which the *ChainOfProvenance* refers.

provenance : ResponsibilityRelationship [1..*]	A set of ResponsibilityRelationships related to the provenance of an entity.
type : ChainOfProvenanceType [0..1]	The type of the *ChainOfProvenance*.

8.4.2 ChainOfProvenanceType

An *ElementType* that specifies a set of expected provenance chains (*ChainOfProvenance*) that capture an ordered set of *ResponsibilityRelationships* of type *ResponsibilityRelationshipType*.

Generalizations

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

Properties

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

Table 40. ChainOfProvenanceType Attributes and/or Associations

Property/Association	Description
entityType : EntityType [1..*]	The *EntityType* for which the *ChainOfProvenanceType* applies.
provenanceType : ResponsibilityRelationshipType [1..*]	The type of the responsibility relationships expected to be included in provenance chains of type *ChainOfProvenanceType*.

8.4.3 ProvenanceChangeKind

A class indicating the kind of provenance change that is expected.

Generalizations

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

Properties

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

8.4.4 ProvenanceChangeOccurrence

An *Occurrence* in the lifecycle of an entity related to the custody and/or ownership of that entity.

Generalizations

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

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

Properties

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

Table 41. ProvenanceChangeOccurrence Attributes and/or Associations

Property/Association	Description
kind : ProvenanceChangeKind [0..1]	A reference to a definition of the specific kind of provenance change.
primaryEntityDependency : OccurrenceDependency [1]	The *OccurrenceDependency* whose `target` is the *Entity* to which the *ProvenanceOccurrence applies.*
priorProvenance : ResponsibilityRelationship [1..*]	The *ResponsibilityRelationships* prior to the *ProvenanceChangeOccurrence*.
responsibleParty : Party [1..*]	The *Party* that has responsibility for the entity as a result of the *ProvenanceOccurrence.*
resultingProvenance : ResponsibilityRelationship [0..*]	The *ResponsibilityRelationships* that result from the *ProvenanceChangeOccurrence*.
subchain : ProvenanceOccurrenceChain [0..1]	A *ProvenanceChain* that is encapsulated by the *ProvenanceOccurrence,* essentially creating a "sub-chain".
type : ProvenanceChangeType [0..1]	The type of the *ProvenanceOccurrence*.

8.4.5 ProvenanceChangeType

The type of a *ProvenanceOccurrence* in the lifecycle of an entity that is of interest to some *Party*.

Generalizations

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

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

Properties

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

Table 42. ProvenanceChangeType Attributes and/or Associations

Property/Association	Description
entityType : OccurrenceDependencyType [1..*]	A relationship to the expected type of entity involved in the *ProvenanceChangeType*.
expectedResponsiblePartyType : PartyType [0..*]	The *Party* that is expected to be responsible in some way for an *entity* of a particular type.
kind : ProvenanceChangeKind [0..1]	A reference to a definition of the specific kind of provenance change.
priorResponsibilityType : ResponsibilityRelationshipType [0..*]	The *ResponsibilityRelationsihipType* exected to exist prior to occurrences of type *ProvenanceChangeType*.

resultingResponsibilityType : ResponsibilityRelationshipType [0..*]	The type of *ResponsibilityRelationships* expected as a result of the *ProvenanceChangeType*.
subchainType : ProvenanceOccurrenceChainType [0..1]	A *ProvenanceChainType* that is encapsulated within the *ProvenanceOccurrenceType* to create a "subchain".

8.4.6 ProvenanceOccurrenceChain

A succession of *ProvenanceOccurrences* that have happened in the life of an entity that is of interest to some *Party*.

Generalizations

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

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

Properties

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

Table 43. ProvenanceOccurrenceChain Attributes and/or Associations

Property/Association	Description
entity : Entity [1]	The entity that is the subject of the *ProvenanceChain*.
occurrenceHistory : ProvenanceChangeOccurrence [0..*]	A set of *ProvenanceOccurrences* that comprise the chain.
type : ProvenanceOccurrenceChainType [0..1]	The type of the *ProvenanceChain*.

8.4.7 ProvenanceOccurrenceChainType

A kind of *OccurrenceChainType* that captures a specification for a series of potential *ProvenanceOccurrences* that are expected in a particular context. A *ProvenanceChainType* captures this specification through the `occurrenceTypeGraph` property - a graph of *OccurrenceGraphNodes* and *OccurrenceTransitionTypes*.

Generalizations

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

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

Properties

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

Table 44. ProvenanceOccurrenceChainType Attributes and/or Associations

Property/Association	Description
entityType : EntityType [1]	The subject of the *ProvenanceChainType*.

occurrenceType : ProvenanceChangeType [1..*]	A derived property that holds the set of *ProvenanceOccurrenceTypes* that represent the types of *ProvenanceOccurrences* expected to occur as part of *ProvenanceChains* that the *ProvenanceChainType* specifies.
occurrenceTypeGraph : ProvenanceTypeGraph [0..1]	A graph of *ProvenanceOccurrenceTypes* that specifies the sequencing of expected *ProvenanceOccurrences* in the lifecycle of an entity of interest to zero or more *InterestedParties*.

8.4.8 ProvenanceTypeGraph

A specialized type of OccurrenceTypeGraph that captures the *ProvenanceOccurrenceTypes* that are expected in the lifecycle of one or more types of entities.

Generalizations

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

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

Properties

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

8.4.9 ResponsibilityRelationship

A *ResponsibilityRelationship* is a kind of *ElementRelationship* that specifies a *Party* has some provenance-related responsibility for an entity for a particular period of time.

Generalizations

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

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

Properties

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

Table 45. ResponsibilityRelationship Attributes and/or Associations

Property/Association	Description
end : DateTime [0..1]	The date on which which a *Party* relinquishes the specified responsibilities with respect to a particular entity.
entity : Entity [1]	The entity for which a *Party* is responsible from either a custody or ownership perspective.
isUnique : Boolean [] default: false	A boolean that indicates whether or not the responsibility is unique.
kind : ResponsibilityRelationshipKind [0..1]	The kind of ResponsibilityRelationship between PartyTypes and EntityTypes in a given situation. See ResponsibilityRelationshipKind for more details.
responsibilityDescription : Documentation [0..1]	A textual description of the responsibility.

responsibleParty : Party [1..*]	The *Party* that is responsible from a provenance perspective for a particular entity.
start : DateTime [0..1]	The date on which a *Party* acquires the responsibilities with respect to a particular entity.
type : ResponsibilityRelationshipType [0..1]	The type of the *ResponsibilityRelationship*.

8.4.10 ResponsibilityRelationshipKind

A class representing the kind of *ResponsibilityRelationship* between *Parties* and *Entities* in some particular situation.

Generalizations

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

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

Properties

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

8.4.11 ResponsibilityRelationshipType

A kind of *ElementRelationshipType* that specifies an expected *ResponsibilityRelationship* between *PartyTypes* and *EntityTypes* in some particular situation.

Generalizations

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

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

Properties

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

Table 46. ResponsibilityRelationshipType Attributes and/or Associations

Property/Association	Description
description : Documentation [0..1]	A textual description of the responsibility.
entityType : EntityType [1..*]	The expected EntityTypes to which the responsibility applies.
kind : ResponsibilityRelationshipKind [1]	A description of the kind of ResponsibilityRelationship between PartyTypes and EntityTypes in a given situation. See ResponsibilityRelationshipKind for more details.
responsiblePartyType : PartyType [1..*]	The *PartyType* that is expected to have the given ResponsibilityRelationshipType with particular EntityTypes in given situations.

8.4.12 Custody

The Custody package provides elements related to the notion of the custody or "physical" control of entities of interest.

PPMN supports tracking the chain of custody of entities of interest. A *ChainOfCustody* tracks the physical or electronic holder of an entity of interest. It does this by referencing a series of *CustodyOccurrences* that represent the custodial history of an entity of interest. A *ChainOfCustody* may have a *ChainOfCustodyType* that defines the *CustodyOccrrenceTypes* expected for a particular *EntityType*.

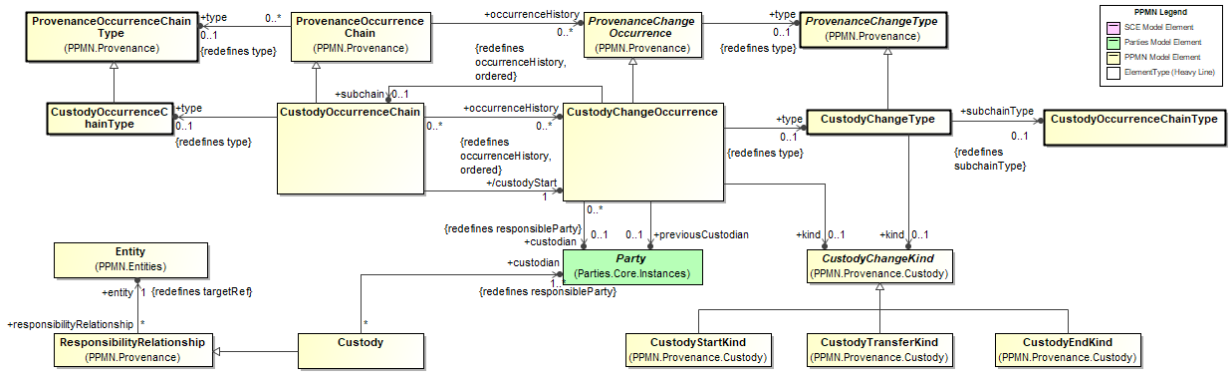


Figure 32: Custody Occurrence Chains

Custody-related classes follow the same pattern that **PPMN** establishes for *Occurrences* generally. This pattern supports the "nesting" of a *ChainOfCustody* within a *CustodyOccurrence*. This pattern allows for encapsulation of parts of a chain where the details of the occurrences of a part of a larger chain are either not known initially or are not deemed important in some context.

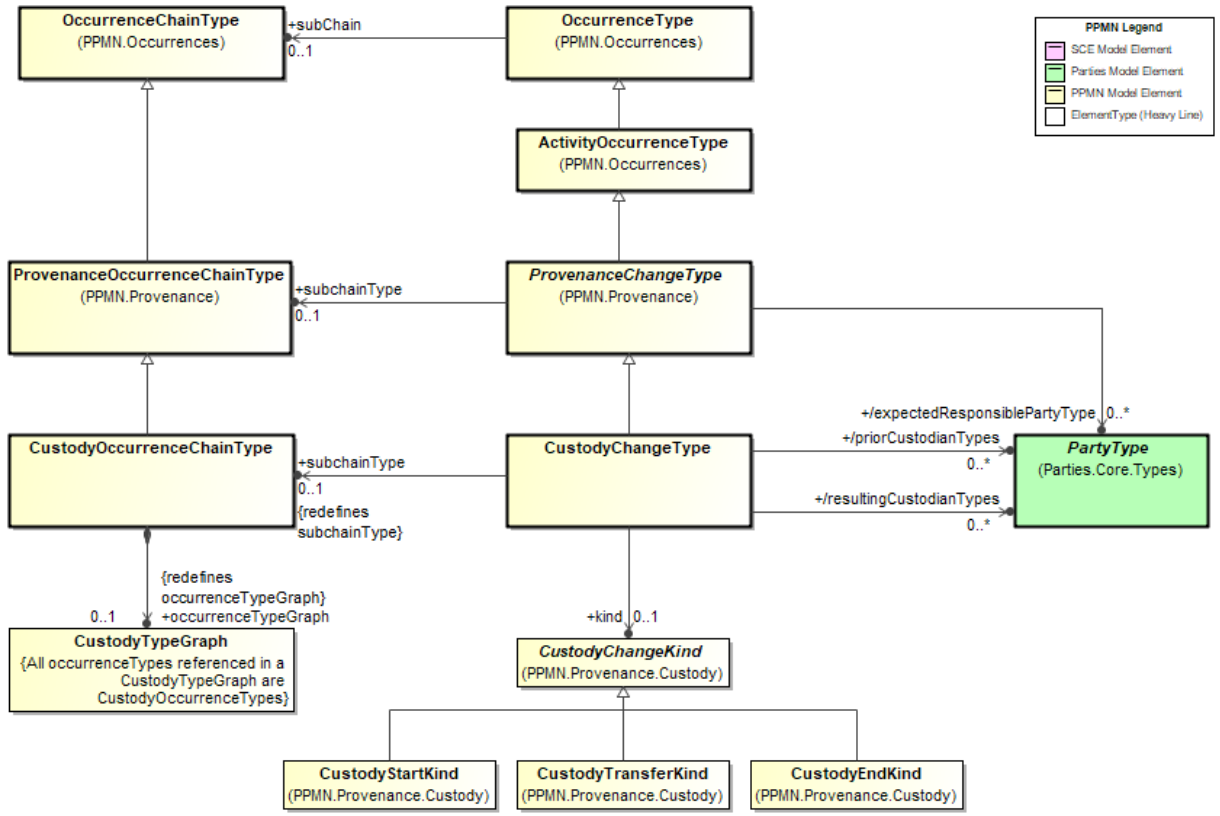


Figure 33: Custody Occurrence Chain Types

Custody-related classes follow the same pattern that **PPMN** establishes for *Occurrences* generally. This pattern supports the "nesting" of a *ChainOfCustody* within a *CustodyOccurrence*. This pattern allows for encapsulation of parts of a chain where the details of the occurrences of a part of a larger chain are either not known initially or are not deemed important in some context.

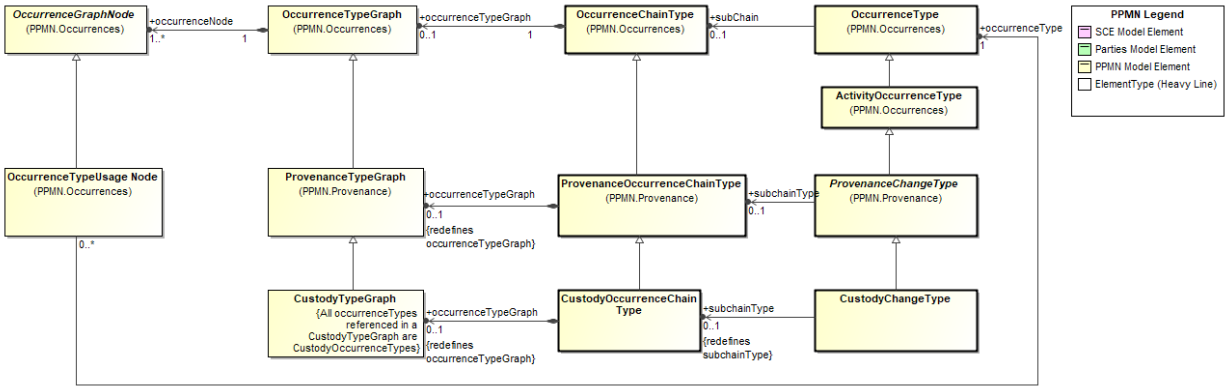


Figure 34: Custody Occurrence Chain Type Pattern

TBD.

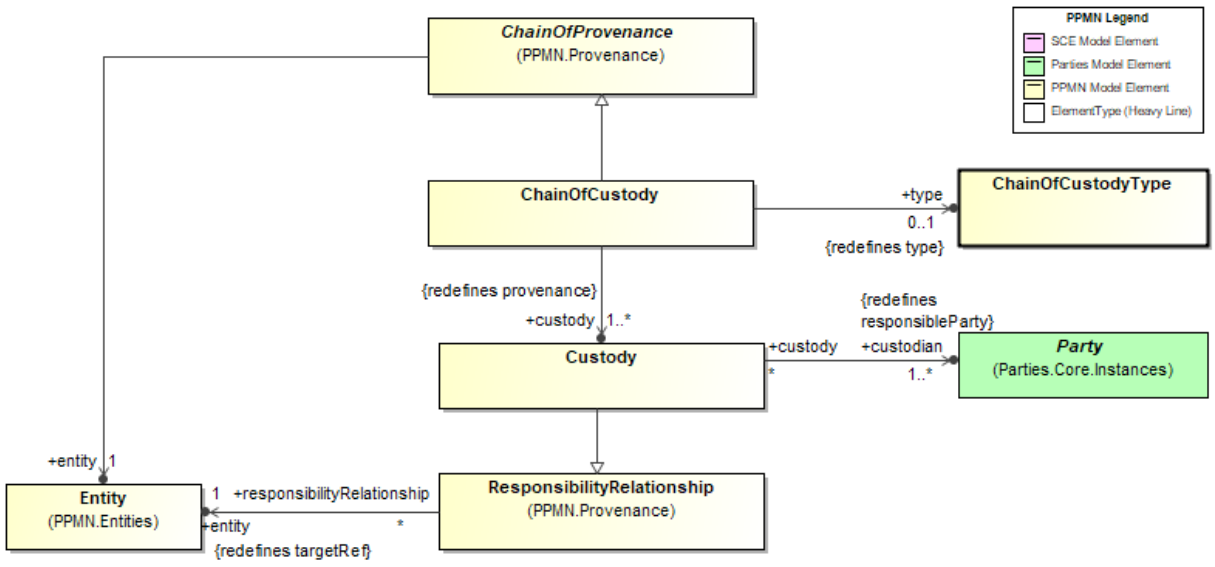


Figure 35: Chain of Custody

TBD.

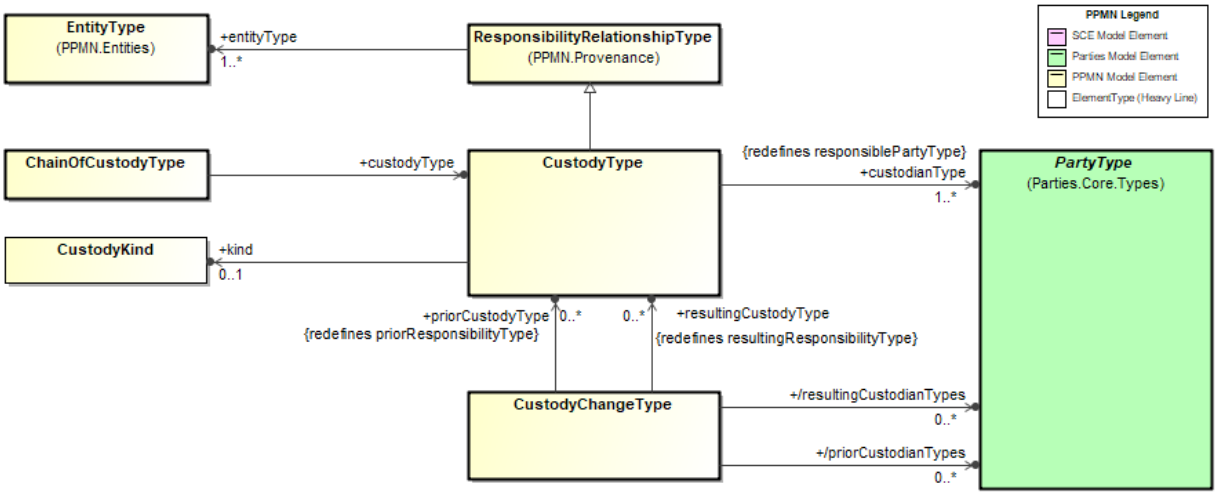


Figure 36: Chain of Custody Types

8.4.12.1 ChainOfCustody

An ordered set of *Custody* relationships that captures the chain of custody of a particular entity over the course of its lifecycle.

Generalizations

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

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

Properties

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

Table 47. ChainOfCustody Attributes and/or Associations

Property/Association	Description
custody : Custody [1..*]	A set of Custody relationships related to the custody of an entity.
type : ChainOfCustodyType [0..1]	The type of the ChainOfCustody.

8.4.12.2 ChainOfCustodyType

A specialization of *ChainOfProvenanceType* that specifies instances of custody chains (*ChainOfCustody*) that capture an ordered set of *Custody* relationships of type *CustodyType.*

Generalizations

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

Properties

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

Table 48. ChainOfCustodyType Attributes and/or Associations

Property/Association	Description
custodyType : CustodyType []	The *CustodyType* of the *Custody* responsibility relationships contained in custody chains of type ChainOfCustodyType.

8.4.12.3 Custody

Custody is a kind of *ProvenanceRecord* that specifies a *Party* that has physical or electronic control of an entity for a particular period of time.

Generalizations

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

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

Properties

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

Table 49. Custody Attributes and/or Associations

Property/Association	Description
custodian : Party [1..*]	The *Party* that acts as the custodian of a particular entity. Redefines `responsibleParty`.

8.4.12.4 CustodyChangeKind

A class indicating the kind of CustodyChangeOccurrence.

Generalizations

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

Properties

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

8.4.12.5 CustodyChangeOccurrence

An occurrence in the lifecycle of an entity related to the custody of that entity.

Generalizations

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

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

Properties

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

Table 50. CustodyChangeOccurrence Attributes and/or Associations

Property/Association	Description
custodian : Party [0..1]	The *Party* that has custody of the entity as a result of the *CustodyChangeOccurrence.*
kind : CustodyChangeKind [0..1]	The kind of .
previousCustodian : Party [0..1]	The *Party* that previously had custody of the entity.
subchain : CustodyOccurrenceChain [0..1]	A *ChainOfCustody* that is encapsulated by the *CustodyChangeOccurrence* essentially creating a "sub-chain".
type : CustodyChangeType [0..1]	The type of the *CustodyChangeOccurrence*.

8.4.12.6 CustodyChangeType

The type of custody-related occurrences in the lifecycle of an entity that is of interest to some *Party.* Specializations of CustodyOccurrence will specify the kind of CustodyOccurrence that has happened or is expected to happen.

Generalizations

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

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

Properties

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

Table 51. CustodyChangeType Attributes and/or Associations

Property/Association	Description
kind : CustodyChangeKind [0..1]	The kind of custody change.

priorCustodianTypes : PartyType [0..*]	The type of *Party* that is expected to relinquish custody of *Entities* of *EntityType* as a result of the *CustodyOccurrence.*
priorCustodyType : CustodyType [0..*]	The *CustodyType* of the *Custody* responsibility relationships expected to be in place prior to *CustodyChangeOccurrences* of type *CustodyChangeType*.
resultingCustodianTypes : PartyType [0..*]	The type of *Party* that is expected to have custody of *Entities* of *EntityType* as a result of the *CustodyOccurrence.*
resultingCustodyType : CustodyType [0..*]	The *CustodyType* expected to be the result of occurrences of type *CustodyChangeType*.
subchainType : CustodyOccurrenceChainType [0..1]	The expected *ChainOfCustodyType* that the *CustodyOccurrenceType* encapsulates.

8.4.12.7 CustodyEndKind

A class indicating the CustodyChangeOccurrence was a kind of end.

Generalizations

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

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

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

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

Properties

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

8.4.12.8 CustodyKind

A class indicating the kind of *Custody* that a *Party* has with respect to some *Entity*.

Generalizations

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

Properties

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

8.4.12.9 CustodyOccurrenceChain

A succession of *CustodyChangeOccurrences* that have happened in the life of an entity that is of interest to some *Party*.

Generalizations

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

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

Properties

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

Table 52. CustodyOccurrenceChain Attributes and/or Associations

Property/Association	Description
custodyStart : CustodyChangeOccurrence [1]	The occurrence that starts the *ChainOfCustody*. This is derived by finding the earliest occurrence in the chain.
occurrenceHistory : CustodyChangeOccurrence [0..*]	A set of *CustodyOccurrences* that comprise the chain.
type : CustodyOccurrenceChainType [0..1]	The type of the *ChainOfCustody*.

8.4.12.10 CustodyOccurrenceChainType

A kind of *ProvenanceChainType* that captures a specification for a series of expected *CustodyOccurrenceTypes* that are expected for a particular entity type.

Generalizations

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

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

Properties

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

Table 53. CustodyOccurrenceChainType Attributes and/or Associations

Property/Association	Description
occurrenceTypeGraph : CustodyTypeGraph [0..1]	A graph of *CustodyOccurrenceTypes* that specifies the sequencing of expected *CustodyOccurrences* in the lifecycle of an entity of interest to one or more *InterestedParties*.

8.4.12.11 CustodyStartKind

A class indicating the CustodyChangeOccurrence was a kind of start.

Generalizations

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

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

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

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

Properties

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

8.4.12.12 CustodyTransferKind

A class indicating the CustodyChangeOccurrence was a kind of transfer.

Generalizations

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

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

Properties

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

8.4.12.13 CustodyType

A specification of the kind of *Custody* that may exist between *Parties* of type *PartyType* and *Entities* of type *EntityType*.

Generalizations

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

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

Properties

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

Table 54. CustodyType Attributes and/or Associations

Property/Association	Description
custodianType : PartyType [1..*]	The PartyType expected to have custodial responsibility.
kind : CustodyKind [0..1]	A specification of the kind of custody responsibility.

8.4.12.14 CustodyTypeGraph

A specialized type of *ProvenanceTypeGraph* that captures the *CustodyOccurrenceTypes* that are expected in the lifecycle of one or more types of entities.

Generalizations

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

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

Properties

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

8.4.13 Ownership

An integral aspect of provenance is ownership - the legal or rightful title to an entity. Ownership is important in that it indicates a legal responsibility for the entity and the right to perform actions on or with the entity in accordance with applicable laws and regulations. The Ownership package of PPMN provides elements related to the notion of the ownership of entities of interest by one or more parties.

OwnershipOccurrences are *Occurrences* that result in some change in ownership such as the acquisition of an entity by some *Party* or the transfer of ownership of an entity from one *Party* to another. These are useful for two reasons.

First, they link ownership "periods" together and provide greater information about the events or processes that result in a transition in ownership much like *PedigreeOccurrences* provide insight into how an entity is created or evolved over time. Second, *Ownership* "records" are generated as a result of *OwnershipOccurrences* and so the *OwnershipOccurrences* provide insight in how and why ownership has changed..

PPMN supports several kinds of *OwnershipOccurrenceTypes*: *AcquisitionOccurrenceTypes*, *OwnershipTransferOccurrenceTypes*, and *EndOwnershipOccurrenceTypes*. These specializations support the typical ownership transitions that may take place in the lifecycle of an entity but are not expected to be only types of transitions that may occur.

A *ChainOfOwnership* is a kind of *ProvenanceChain* that tracks the ownership-related *Occurrences* of an entity of interest. A *ChainOfOwnership* may be typed in the same way as *ProvenanceChains* using a *ChainOfOwnershipType*. *ChainOfOwnershipType* allows stakeholders to define the expected changes in ownership of entities of a particular type in advance for planning or other purposes.

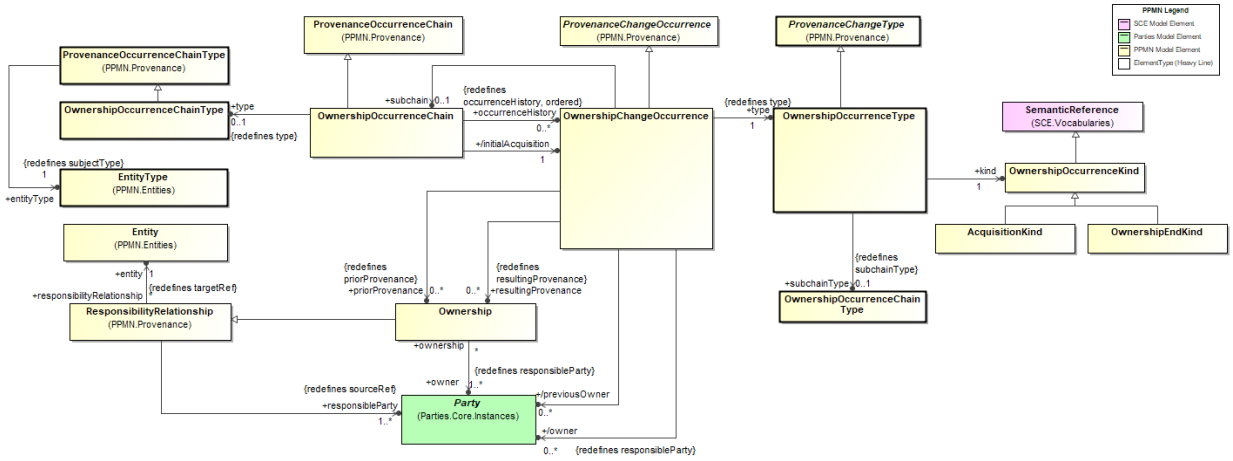


Figure 37: Ownership Occurrence Chains

ChainOfOwnership, *ChainOfOwnershipType*, *OwnershipOccurrences*, and *OwnershipOccurrenceTypes* follow the same pattern established for other types of occurrences. This pattern supports the "nesting" of a *ChainOfOwnership* within an *OwnershipOccurrence*. This pattern allows for encapsulation of parts of a chain where the details of the *OwnershipOccurrences* of that part of a larger chain are either not known initially or are not deemed important in some context.

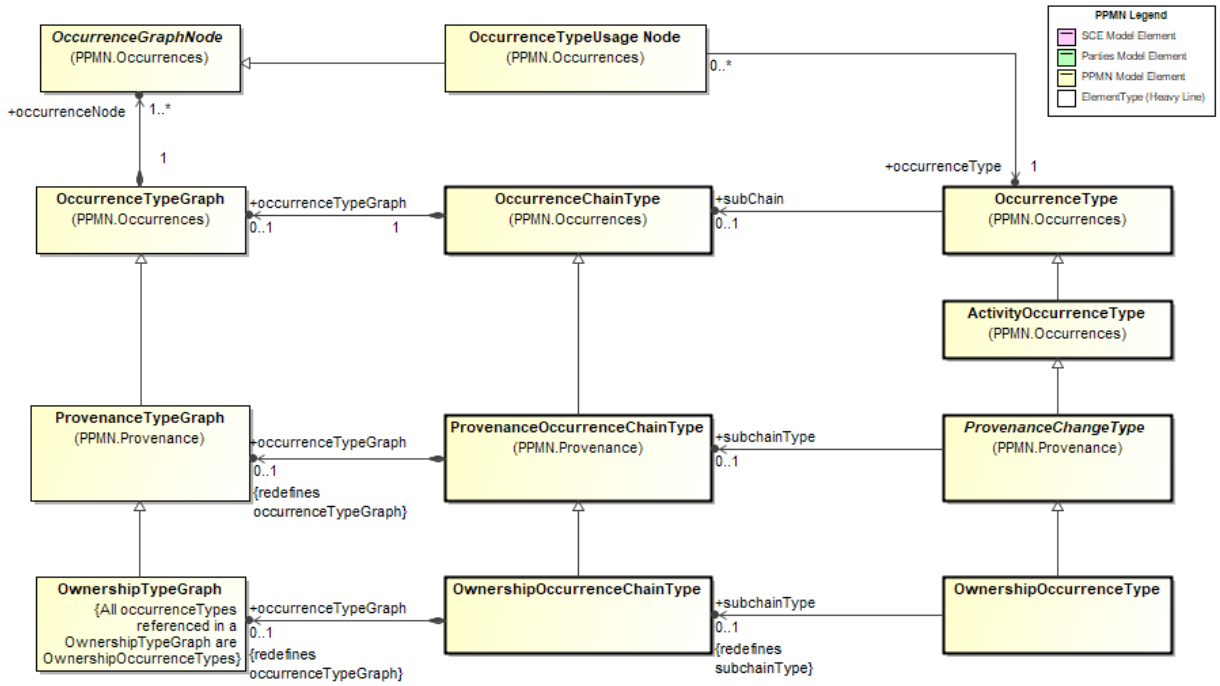


Figure 38: Ownership Occurrence Chain Type Pattern

ChainOfOwnership, *ChainOfOwnershipType*, *OwnershipOccurrences*, and *OwnershipOccurrenceTypes* follow the same pattern established for other types of occurrences. This pattern supports the "nesting" of a *ChainOfOwnership* within an *OwnershipOccurrence*. This pattern allows for encapsulation of parts of a chain where the details of the *OwnershipOccurrences* of that part of a larger chain are either not known initially or are not deemed important in some context.

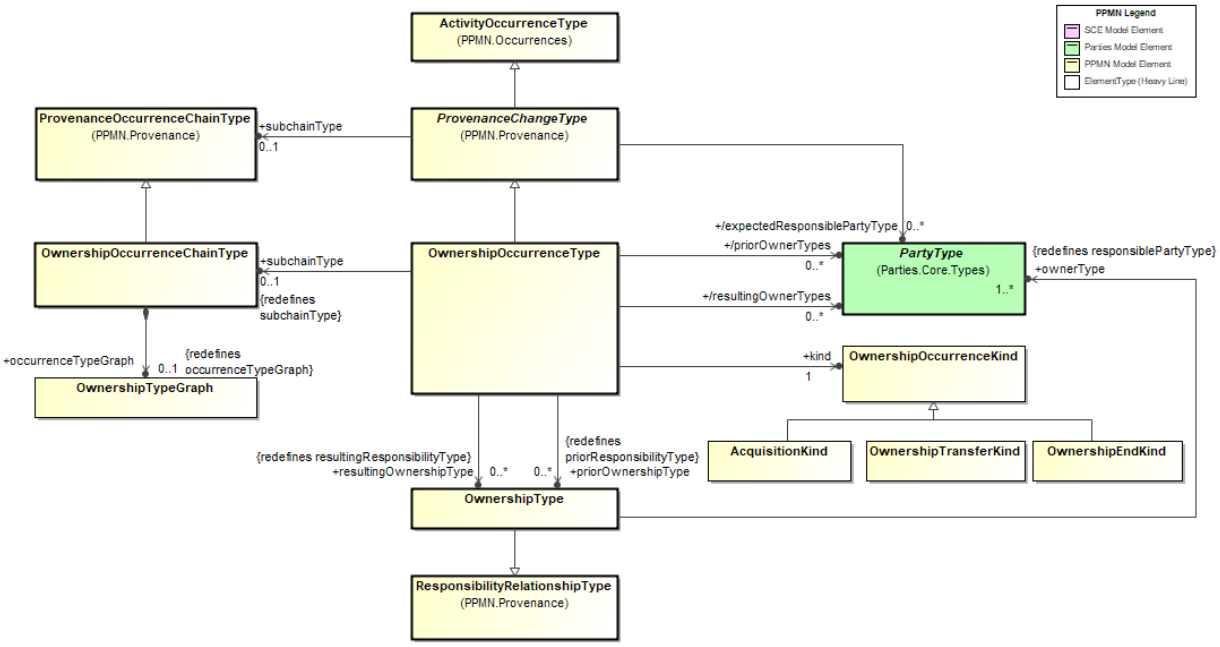


Figure 39: Ownership Occurrence Chain Types

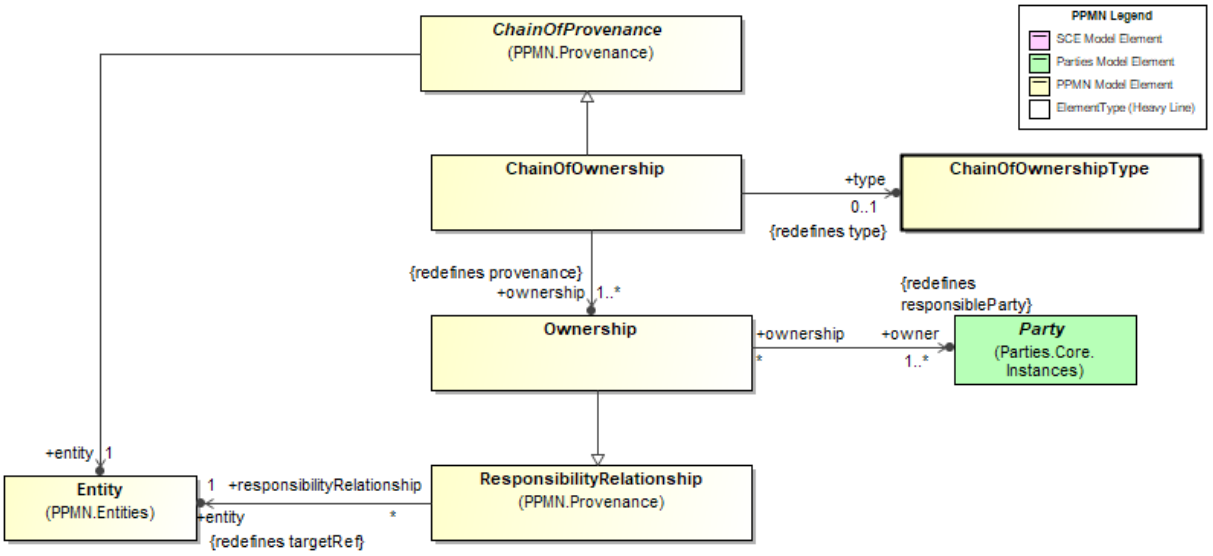


Figure 40: Chain of Ownership

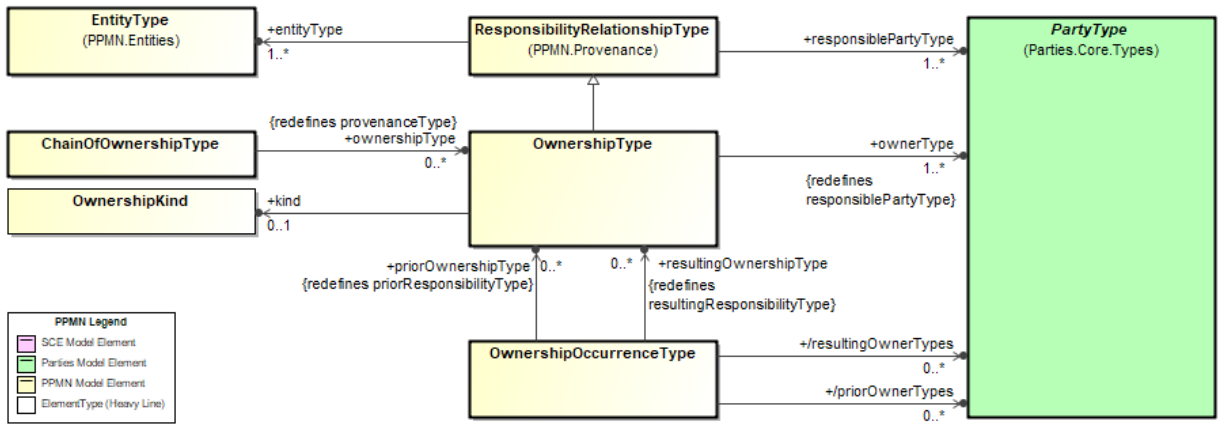


Figure 41: Chain of Ownership Types

8.4.13.1 AcquisitionKind

A class indicating how a *ChainOfOwnership* was started.

Generalizations

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

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

Properties

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

8.4.13.2 ChainOfOwnership

An ordered set of *Ownership* relationships that captures the ownership of a particular entity over the course of its lifecycle.

Generalizations

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

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

Properties

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

Table 55. ChainOfOwnership Attributes and/or Associations

Property/Association	Description
ownership : Ownership [1..*]	A set of Ownership relationships related to the ownership of an entity.
type : ChainOfOwnershipType [0..1]	The type of the ChainOfOwnership.

8.4.13.3 ChainOfOwnershipType

A specialization of *ChainOfProvenanceType* that specifies instances of ownership chains (*ChainOfOwnership*) that capture an ordered set of *Ownership* relationships of type *OwnershipType.*

Generalizations

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

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

Properties

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

Table 56. ChainOfOwnershipType Attributes and/or Associations

Property/Association	Description
ownershipType : OwnershipType [0..*]	The *OwnershipType* of the *Ownership* responsibility relationships included in *ChainOfOwnershps* that are of type *ChainOfOwnershipType*.

8.4.13.4 Ownership

A kind of *ProvenanceRecord* relationship that specifies a *Party* is playing the role of *Owner* of an entity for a particular period of time.

Generalizations

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

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

Properties

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

Table 57. Ownership Attributes and/or Associations

Property/Association	Description
owner : Party [1..*]	The *Party* that acts as the owner of a particular entity. Redefines `responsibleParty`.

8.4.13.5 OwnershipChangeOccurrence

An *Occurrence* in the lifecycle of an entity related to the ownership of that entity.

Generalizations

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

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

Properties

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

Table 58. OwnershipChangeOccurrence Attributes and/or Associations

Property/Association	Description
owner : Party [0..*]	The *Party* that has ownership of the entity as a result of the *OwnershipOccurrence.*
previousOwner : Party [0..*]	The previous owner(s) of the entity.
priorProvenance : Ownership [0..*]	The *Ownership* relationships prior to the *OwnershipChangeOccurrence*.
resultingProvenance : Ownership [0..*]	The *Ownership* relationships that result from the *OwnershipChangeOccurrence*.
subchain : OwnershipOccurrenceChain [0..1]	A *ChainOfOwnership* that is encapsulated by the *OwnershipOccurrence* essentially creating a "sub-chain".
type : OwnershipOccurrenceType [1]	The type of the *OwnershipChangeOccurrence*.

8.4.13.6 OwnershipEndKind

A class indicating how the *ChainOfOwnership* was ended.

Generalizations

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

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

Properties

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

8.4.13.7 OwnershipKind

A specification of a particular kind of ownership responsibility.

Generalizations

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

Properties

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

8.4.13.8 OwnershipOccurrenceChain

A succession of *OwnershipOccurrences* that have happened in the life of an entity that is of interest to some *Party*.

Generalizations

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

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

Properties

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

Table 59. OwnershipOccurrenceChain Attributes and/or Associations

Property/Association	Description
initialAcquisition : OwnershipChangeOccurrence [1]	The occurrence that starts the *ChainOfOwnership*. This is derived by finding the earliest occurrence in the chain.
occurrenceHistory : OwnershipChangeOccurrence [0..*]	A set of *OwnershipOccurrences* that comprise the chain.
type : OwnershipOccurrenceChainType [0..1]	The type of the *ChainOfOwnership*.

8.4.13.9 OwnershipOccurrenceChainType

A kind of *ProvenanceChainType* that captures a specification for a series of expected *OwnershipOccurrenceTypes* that are expected for a particular entity type. An *OwnershipOccurrenceType* captures this specification through the `occurrenceTypeGraph` property - a graph of *OccurrenceGraphNodes* and *OccurrenceTransitionTypes*.

Generalizations

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

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

Properties

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

Table 60. OwnershipOccurrenceChainType Attributes and/or Associations

Property/Association	Description
occurrenceTypeGraph : OwnershipTypeGraph [0..1]	A graph of *OwnershipOccurrenceTypes* that specifies the sequencing of expected *OwnershipOccurrences* in the lifecycle of an entity of interest to one or more *InterestedParties*.

8.4.13.10 OwnershipOccurrenceKind

A class indicating the kind of *OwnershipOccurrence* that is expected.

Generalizations

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

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

Properties

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

8.4.13.11 OwnershipOccurrenceType

The type of *OwnershipOccurrence* in the lifecycle of an entity that is of interest to some *Party*. *S*pecializations of *OwnershipOccurrenceType* will specify the kind of *OwnershipOccurrence* that has happened.

Generalizations

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

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

Properties

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

Table 61. OwnershipOccurrenceType Attributes and/or Associations

Property/Association	Description
kind : OwnershipOccurrenceKind [1]	A reference to a definition of the specific kind of *OwnershipOccurrenceType*.
priorOwnershipType : OwnershipType [0..*]	The *OwnershipType* exected to exist prior to occurrences of type *OwnershipOccurrenceType*.
priorOwnerTypes : PartyType [0..*]	The type of *Party* that is expected to relinquish ownership of *Entities* of *EntityType* as a result of the *OwnershipOccurrence.*
resultingOwnershipType : OwnershipType [0..*]	The *OwnershipType* expected to be the result of occurrences of type *OwnershipOccurrenceType*.
resultingOwnerTypes : PartyType [0..*]	The type of *Party* that is expected to have owership of *Entities* of *EntityType* as a result of *Occurrences* of the *OwnershipOccurrenceType.*

subchainType : OwnershipOccurrenceChainType [0..1]	A *ChainOfOwnershipType* that is encapsulated within the *OwnershipOccurrenceType* to create a "subchain".

8.4.13.12 OwnershipTransferKind

A class indicating how a *ChainOfOwnership* was started.

Generalizations

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

Properties

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

8.4.13.13 OwnershipType

The type of *Ownership* that may exist between *Parties* of type *PartyType* and *Entities* of type *EntityType*.

Generalizations

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

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

Properties

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

Table 62. OwnershipType Attributes and/or Associations

Property/Association	Description
kind : OwnershipKind [0..1]	A specification of the kind of ownership responsibility.
ownerType : PartyType [1..*]	The PartyType expected to have ownership responsibility.

8.4.13.14 OwnershipTypeGraph

A specialized type of *ProvenanceTypeGraph* that captures the *OwnershipOccurrenceTypes* that are expected in the lifecycle of one or more types of entities.

Generalizations

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

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

Properties

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

8.5 Claims

The Claims package contains elements related to C*laims* made by *Parties* about *Occurrences*.

In many situations, pedigree and/or provenance information about entities is put forth by some party as being true when in fact, that information may be disputed and even shown to be false. *Claims* provide a mechanism to note the *Party* (the claimant) that claims an *Occurrence* has happened. The time the claim was made is captured as well as whether the *Claim* was made in a "positive" or "negative" manner (the `claimPositivity`). ClaimPositivity states whether the Claim was made in a "positive" manner, i.e., the Occurrence is claimed to have happened, or a "negative" maner, i.e., the Occurrence is claimed *not to have happened.* A `claimPositivity` of "Possible" means that the Occurrence *may* have happened.

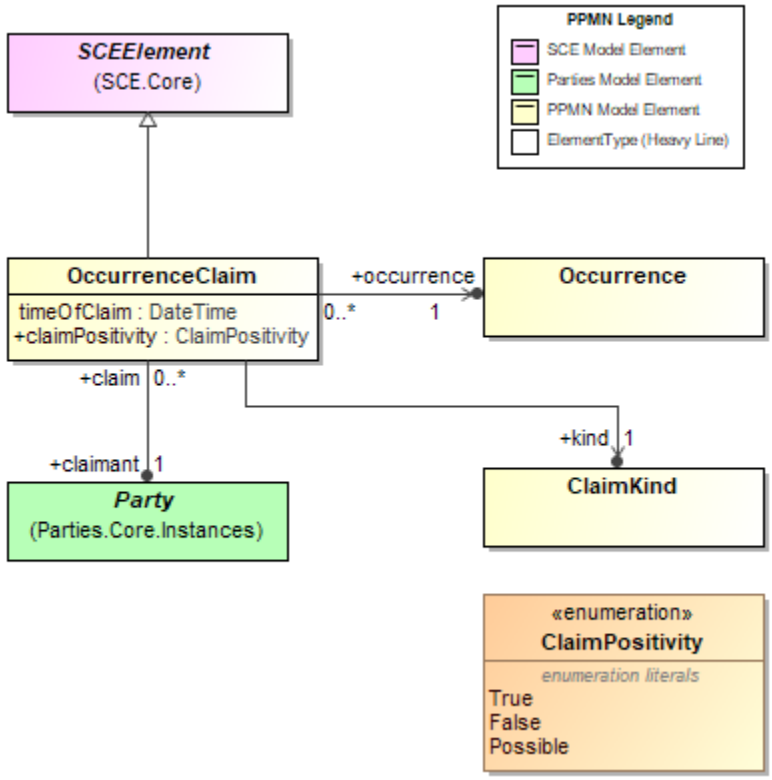


Figure 42: Claims

A Claim may be assessed in some way as stated by a ClaimAssessment by some Party (the assessor). The actual method or mechanism of the assessment is outside the scope of this specification.

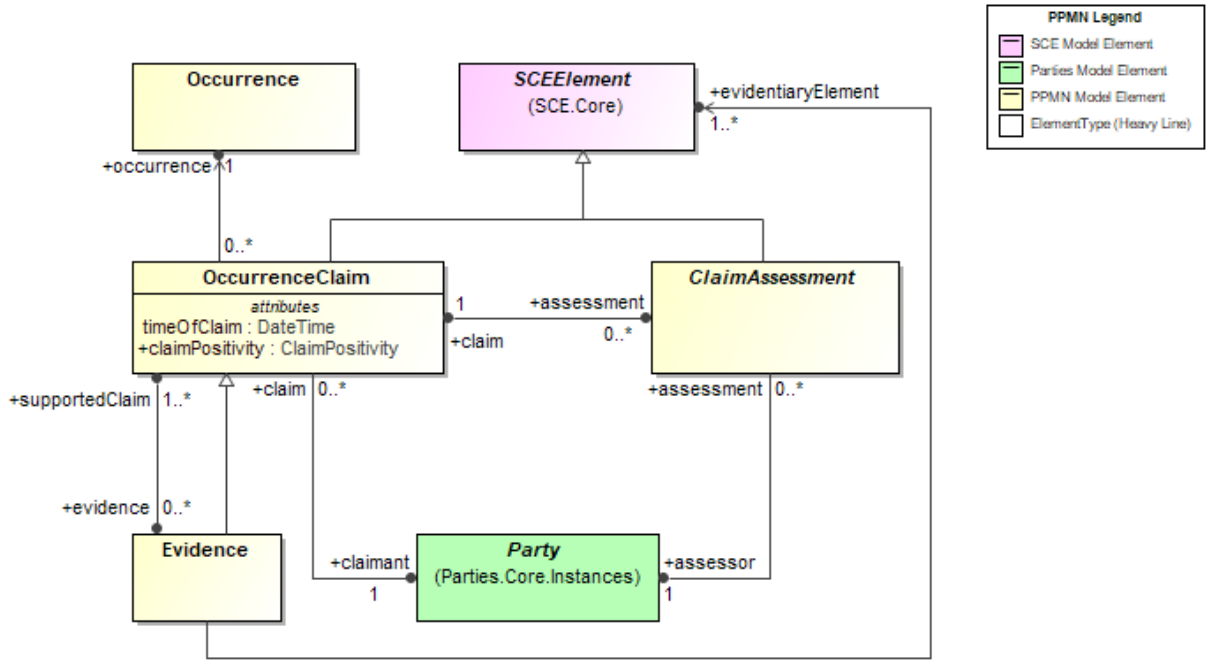


Figure 43: Claim Assessments

8.5.1 ClaimPositivity

A enumeration that indicates whether the statement asserted by a Claim is asserted as being true, false, or possible.

Table 63. ClaimPositivity Literals

Literal	Description
False	Indicates that the Claim asserts the Occurrence did not happen.
Possible	Indicates that the Claim asserts the Occurrence may have happened.
True	Indicates that the Claim asserts the Occurrence happened.

8.5.2 ClaimAssessment

An assessment of a Claim by an assessor.

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 64. ClaimAssessment Attributes and/or Associations

Property/Association	Description
assessor : Party [1]	The Party that made the assessment.
claim : OccurrenceClaim [1]	The Claim about which the assessment was made.

8.5.3 ClaimKind

A class that indicates the kind of *Claim* that has been made.

Generalizations

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

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

Properties

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

Generalizations

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

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

Properties

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

Table 65. Evidence Attributes and/or Associations

Property/Association	Description
evidentiaryElement : SCEElement [1..*]	The elements that comprise the *Evidence* for the supported *Claims*.
supportedClaim : OccurrenceClaim [1..*]	The *Claims* that the *Evidence* is intended to support.

8.5.4 OccurrenceClaim

A statement made by a Party about whether an Occurrence happened or not.

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 66. OccurrenceClaim Attributes and/or Associations

Property/Association	Description
assessment : ClaimAssessment [0..*]	An assessment of the Claim.
claimant : Party [1]	The Party that made the Claim.
claimPositivity : ClaimPositivity []	A property that states whether the claim is said to be true, false or possible.
evidence : Evidence [0..*]	The *Evidence* intended to support the *Claim*.
kind : ClaimKind [1]	The kind of assertion of the Claim.
occurrence : Occurrence [1]	The Occurrence about which the Claim was made.
timeOfClaim : DateTime []	The time the Claim was made.

8.6 Rationale

The Rationale package contains elements that provide the ability to capture the rationale for *Occurrences*.

PPMN supports the ability to capture a *Rationale*, the reasoning or justification, for *Occurrences* and *OccurrenceTypes*. *RationaleType* enables capture of the type of a particular Rationale or of the kind of *Rationale* that is expected in a particular context.

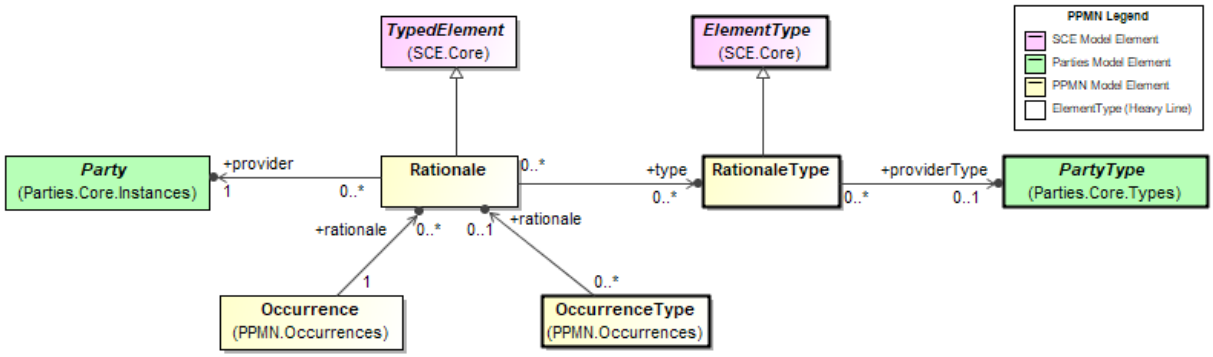


Figure 44: Rationale

8.6.1 Rationale

A class representing the basis for an *Occurrence or OccurrenceType*.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

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

Table 67. Rationale Attributes and/or Associations

Property/Association	Description
provider : Party [1]	The *Party* that provided the *Rationale*.
type : RationaleType [0..*]	The class(es) that provide(s) a specification of the *Rationale*.

8.6.2 RationaleType

A class representing the type or classification of a *Rationale*.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

Table 68. RationaleType Attributes and/or Associations

Property/Association	Description
providerType : PartyType [0..1]	The *PartyType* that is expected to provide the kind of *Rationale* specified by the *RationaleType*.

8.7 Extensions

PPMN includes two mechanisms for extension: Adornments and Annotations. Descriptions of these two mechanisms are described herein.

8.7.1 Adornment

The Adornment package contains elements that support the extension of elements with addtional attributes using the adornment pattern.

PPMN *AdornmentProfiles* extend the **SCE** extension mechanism to allow for the addition of attributes to any *BaseElement* in Pedigree and Provenance information without having to modify that element. The approach is analogous to the Gang of Four adornment design pattern wherein additional features are added to elements without those extensions having to be known when the original element is created. The **SCE** extension mechanism allows for *BaseElements* include extension attributes and values that have been defined by a tool that implements the **SCE** specification. The attributes become part of the BaseElement. **PPMN** *AdornmentProfiles* provide the additional ability to "adorn" the elements with attributes.

PPMN *AdornmentProfile* extends the **SCE** extension mechanism with a number of key features. *AdornmentProfiles* specialize *ExtensionDefinition* to include a version number, a set of *AdornedElements*, and a set of *AdornmentAttributeDefinitions*. The *AdornedElements* referenced by the *AdornmentProfile* specify which *AdornmentAttributeDefinitions* may adorn which *BaseElements*. The *AdornmentProfile's* set of *attributeDefinitions* are an additional set of definitions that may be applied generally rather than to specific *BaseElements*.

AdornmentAttributeDefinition extends **SCE** *ExtensionAttributeDefinition* to provide additional detail about the characteristics of the adornment attributes. *AdornmentValues*, as specified by *AdornmentAttributeDefinition*, may be chronicled (successive versions are tracked), removable or not, modifiable or not, and required at creation of the

BaseElement to which they are applied. *AdornmentAttributeDefinition* also includes a pointer to an *AttributeType* that specifies whether the *AttributeValue* will be an integer, a string, or a date/time.

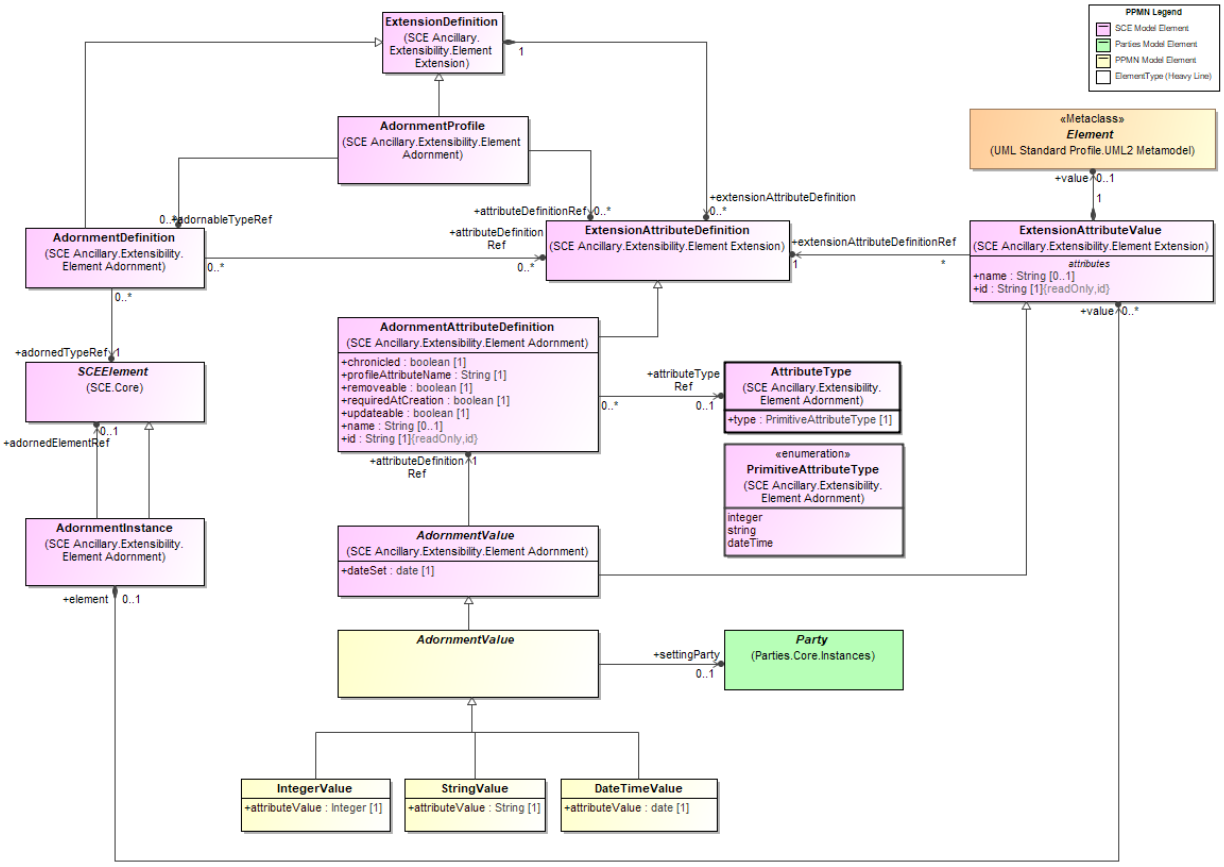


Figure 45: Adornment Profiles

8.7.1.1 AdornmentValue

A value of an attribute associated with an AdornedElement. PPMN AdornmentValue is specialization of SCE AdornmentValue that extends the SCE AdornmentValue to include the party that set the value.

Generalizations

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

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

Properties

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

Table 69. AdornmentValue Attributes and/or Associations

Property/Association	Description
settingParty : Party [0..1]	The *Party* that set the adornment value.

8.7.1.2 DateTimeValue

An AdornmentValue that is an DateTime type.

Generalizations

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

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

Properties

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

Table 70. DateTimeValue Attributes and/or Associations

Property/Association	Description
attributeValue : date [1]	The actual value of the *DataTimeValue*.

8.7.1.3 IntegerValue

An AdornmentValue that is an Integer type.

Generalizations

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

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

Properties

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

Table 71. IntegerValue Attributes and/or Associations

Property/Association	Description
attributeValue : Integer [1]	The actual value of the *IntegerValue*.

8.7.1.4 StringValue

An AdornmentValue that is a String type.

Generalizations

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

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

Properties

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

Table 72. StringValue Attributes and/or Associations

Property/Association	Description
attributeValue : String [1]	The actual value of the *StringValue*.

8.7.2 Annotations

The Annotation package contains elements related to the notion of annotation of elements with notes about that element.

Annotations are applied to *NamedElements* for any purpose that suits the business needs of an organization. *Annotations* can exist independently of those elements providing a “catalog” of *Annotations*. *AnnotationTemplate* provides a means of creating base annotations that can be "instantiated" as either *SimpleAnnotations* or *ChronicledAnnotations*. *Annotations* may have an association to the *AnnotationTemplate* from which they were created. The *Party* creating an *Annotation* is captured as the creator. That *Party* or another *Party* may assign an annotation to a *NamedElement* through an *AnnotationAssignment* relationship.

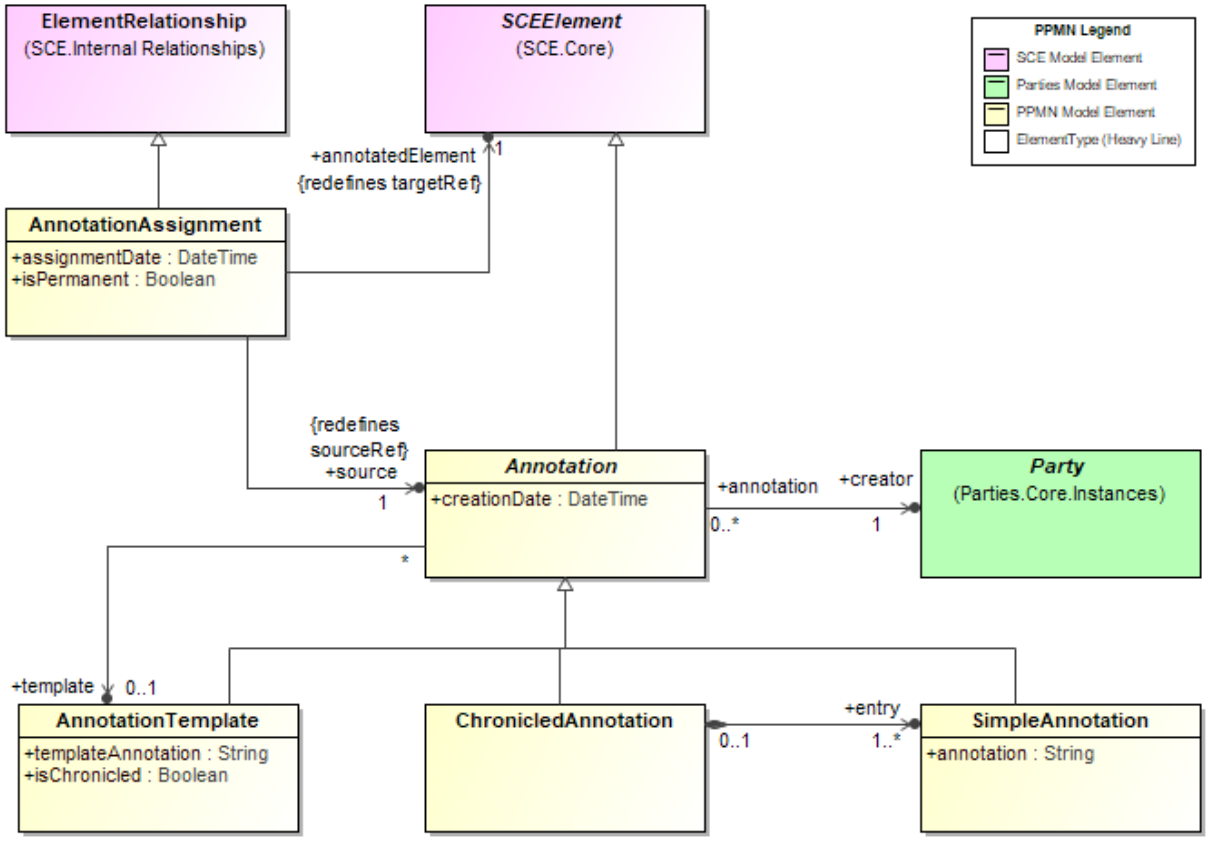


Figure 46: Annotations

8.7.2.1 Annotation

A note or series of notes related to some NamedElement in a PPM information set.

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 73. Annotation Attributes and/or Associations

Property/Association	Description
creationDate : DateTime []	The Date/Time that the Annotation was created.
creator : Party [1]	The *Party* that created the annotation.
template : AnnotationTemplate [0..1]	The template from which an *Annotation* was created.

8.7.2.2 AnnotationAssignment

An association that links an *Annotation* to a *NamedElement* in a PPMN information set.

Generalizations

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

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

Properties

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

Table 74. AnnotationAssignment Attributes and/or Associations

Property/Association	Description
annotatedElement : SCEElement [1]	The element to which the *Annotation* has been assigned.
assignmentDate : DateTime []	The Date/Time the *Annotation* was applied.
isPermanent : Boolean []	A boolean specifying whether or not the *Annotation* is intended to be permanent.
source : Annotation [1]	The *Annotation* that has been assigned to some element.

8.7.2.3 AnnotationTemplate

A kind of *Annotation* that is intended to be used as a template for other *Annotations*.

Generalizations

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

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

Properties

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

Table 75. AnnotationTemplate Attributes and/or Associations

Property/Association	Description
isChronicled : Boolean []	A boolean that specifies whether the *Annotations* created with this template are *ChronicledAnnotations* or not.

templateAnnotation : String []	A default string that is meant for recurring use.

8.7.2.4 ChronicledAnnotation

A kind of *Annotation* that has a series of time-based entries. Individual entries are captured as *SimpleAnnotations* with the `isPermenant` flag set to True. The `creationDate` of the *SimpleAnnotations* that represent the entries of a *ChronicledAnnotation* captures the date the *ChronicledAnnotation* was updated.

Generalizations

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

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

Properties

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

Table 76. ChronicledAnnotation Attributes and/or Associations

Property/Association	Description
entry : SimpleAnnotation [1..*]	A *SimpleAnnotation* that represents one entry in a *ChronicledAnnotation*.

8.7.2.5 SimpleAnnotation

A kind of *Annotation* that is a simple note related to one or more *NamedElements* in a PPM information set.

Generalizations

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

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

Properties

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

Table 77. SimpleAnnotation Attributes and/or Associations

Property/Association	Description
annotation : String []	A string containing the text of the *Annotation.*

8.8 Delegation

The Delegation package provides elements related to the notion of delegation of responsiblities for an entity from one party to another.

Delegation captures the notion that a Party may assign a set of responsibilities to another party. The responsibilities being assigned are essentially captured as a Role. The class ActedOnBehalfOf is a relationship that states that one Party was acting for or representing another Party and that action may be justified by a Delegation. The property `inRole` allows a model to specify that the *Party* acted on behalf of another *Party* while performing a particular role in an *Occurrence*.

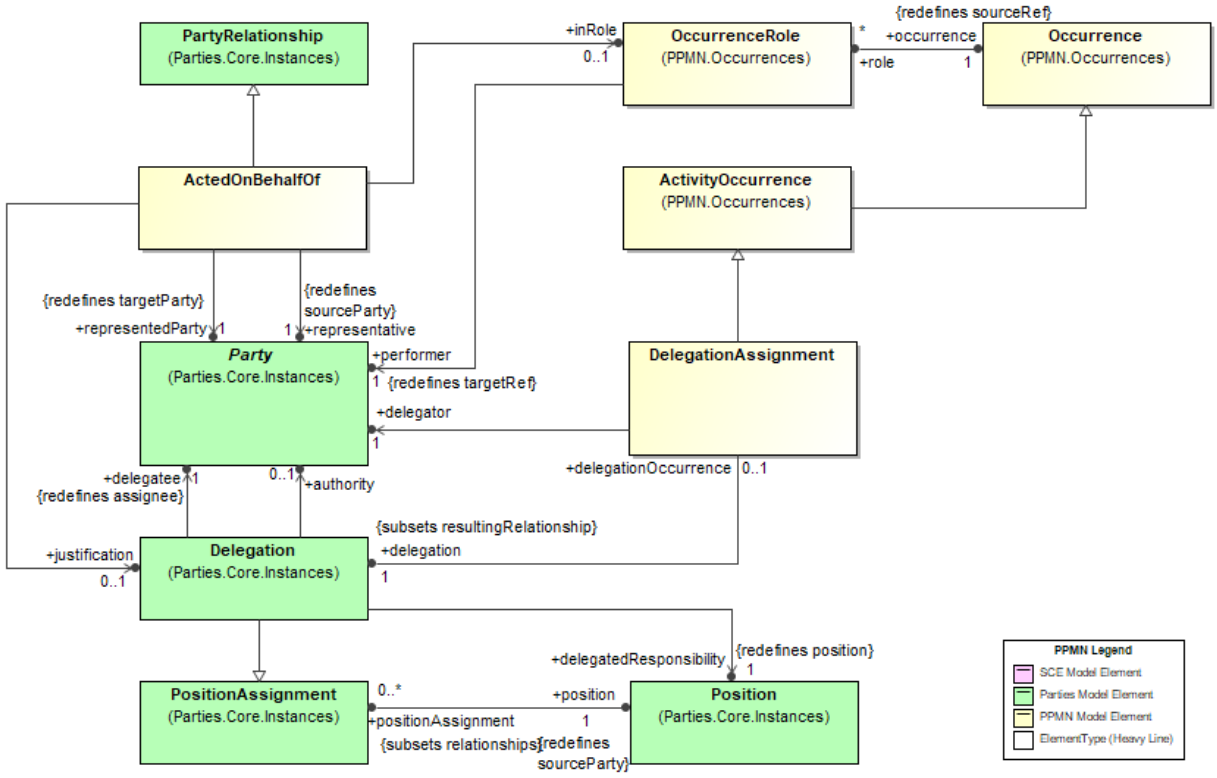


Figure 47: Delegation

8.8.1 ActedOnBehalfOf

A relationship that indicates that one *Party* represented another *Party* in some way. That action may be justified by some *Delegation* of responsibilities.

Generalizations

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

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

Properties

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

Table 78. ActedOnBehalfOf Attributes and/or Associations

Property/Association	Description
inRole : OccurrenceRole [0..1]	The *OccurrenceRole* in which one *Party* acted on behalf of another *Party*.
justification : Delegation [0..1]	The *Delegation* that provides justification for the `representative` to act on the part of the `representedParty`.
representative : Party [1]	The *Party* representing the `representedParty`.

representedParty : Party [1]	The *Party* on whose part the `representative` acted.

8.8.2 DelegationAssignment

A kind of *ActivityOccurrence* wherein one *Party* delegates a set of responsibilities to another *Party*.

Generalizations

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

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

Properties

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

Table 79. DelegationAssignment Attributes and/or Associations

Property/Association	Description
delegation : Delegation [1]	The *Delegation* that was the result of the *DelegationAssignment*.
delegator : Party [1]	The *Party* responsible for the *DelegationAssignment*.

8.9 Additional Relationships

In addition to Delegation and Derivation, PPMN includes a number of other types of relationships that are important to pedigree and/or provenance. These additional relationships are described herein.

PPMN includes several other types of relationships that may be important to particular stakeholders in addition to derivations and delegations. These cover the concepts of attribution, specialization, alternates and general "informing of".

Attribution is captured through the *AttributedTo* relationship. This element states that an entity of interest was generated through some unknown activity or action of the *Party*.

The *Specializes* relationship specifies that one element represents a more specific type of thing than the target of the relationship. The *Specializes* relationship will generally between two entities of some type. However, this is not mandated. It may be useful in certain situations to note specialization relationships between *Parties* or *Occurrences*. Note that the `source` and `target` of a *Specialization* must both be of the same general "type". In other words they must both be, for example, *Entities*, or both be *Parties*, or both be *Occurrences*.

The *AlternateOf* relationship states that two entities or elements represent the same thing or aspects of the same thing. The *AlternateOf* relationship will generally between two entities of some type. As with *Specializes*, however, this is not always the case. It may be useful in certain situations to note alternate *Parties* or *Occurrences*. Note that the `source` and `target` of the *AlternateOf* must both be of the same general "type". In other words they must both be, for example, *Entities*, or both be *Parties*, or both be *Occurrences*.

The *Informed* relationship is used to show that one *Occurrence* provided information or insight to or in some way affected another *Occurrence*. For example a testing process may inform a redesign of an assembly line for a manufacturer.

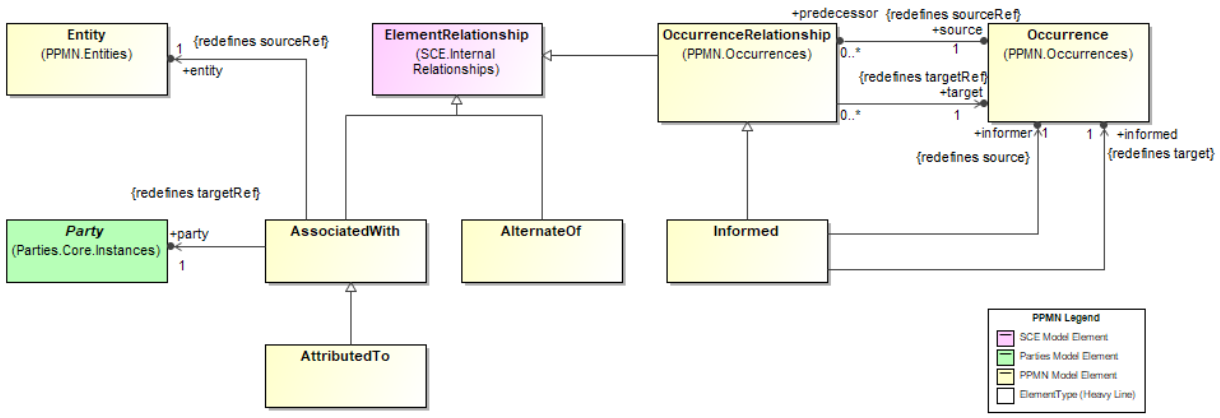


Figure 48: Additional PPMN Relationships

8.9.1 AlternateOf

The *AlternateOf* relationship is a kind of *ElementRelationship* that states that two elements represent the same thing or aspects of the same thing. The *AlternateOf* relationship will generally between two entities of some type. As with *Specializes*, however, this is not always the case. It may be useful in certain situations to note alternate *Parties* or *Occurrences*. Note that the `source` and `target` of the *AlternateOf* must both be of the same general "type". In other words they must both be, for example, *Entities*, or both be *Parties*, or both be *Occurrences*.

Generalizations

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

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

Properties

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

8.9.2 AssociatedWith

The *AssociatedWith* relationship is a kind of *ElementRelationship* that captures the fact that a *Party* is associated in some way with an *Entity*.

Generalizations

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

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

Properties

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

Table 80. AttributedTo Attributes and/or Associations

Property/Association	Description
entity : Entity [1]	An entity that is associated with some *Party*.
party : Party [1]	The *Party* to which some entity is associated.

8.9.3 AttributedTo

The *AttributedTo* relationship is a kind of *AssociatedWith* relationship that captures the fact that an Entity was created or transformed by some unknown activity or action of a *Party*.

Generalizations

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

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

8.9.4 Informed

The *Informed* relationship is a kind of *ElementRelationship* that is used to show that one *Occurrence* provided information or insight to or in some way affected another *Occurrence*.

Generalizations

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

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

Properties

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

Table 81. Informed Attributes and/or Associations

Property/Association	Description
informed : Occurrence [1]	The *Occurrence* that was informed by the source *Occurrence*.
informer : Occurrence [1]	The *Occurrence* that informed another *Occurrence*.

8.10 Packaging

PPMN Packaging consists of elements that allow users to group or "package up" sets of occurrences associated with the pedigree and provenance of entities of interest as well as elements that define expected occurrences. The packaging follows the pattern laid out in the Specification Common elements (SCE) specification and used in the Parties specification as well.

The Pedigree and Provenance Metamodel and Notation supports the capture of events that happen in the lifecycle of entities of interest including creation, evolution, destruction, as well as changes in ownership and custody. In addition to capturing events that happened in the past, the specification also enables specifying events that are expected to happen in the future. As stated previously, these elements are loosely referred to as the "instances" and "types", respectively. The main packaging structures of PPMN support packaging of these elements using *PPMNInstances* and the *PPMNDefinitions* elements.

PPMNInstances are specializations of *PartyInstances* and are designed to group "instances" related to events that have taken place in the lifecycle of entities of interest. These elements include actual events or *Occurrences*, the *Entities*, and the *Parties* involved.

PPMNDefinitions are specializations of *PartyDefinitions* and are designed to group the **PPMN** "types", i.e. the elements related to "expected" *Occurrences*. These elements include *OccurrenceTypes*, *EntityTypes*, and *PartyTypes* among others. *PPMNDefinitions* also reference any profiles that have been applied through the `appliedProfile` property. Any applied profiles must be contained via the inherited `profile` property.

PPMNInstances and PPMNDefinitions together are included in PPMNModels along with relevant *PPMNVocabularies*. *PPMNModels* represent the semantics of the model versus the presentation elements contained

in the *PPMNDI* package. *PPMNModels* are specializations of *PartyModels* and so may include *PartyInstances* and *PartyDefinitions* as well.

All of these elements are brought together as a complete bundle in the *PPMNModelPackage*. *PPMNModelPackages* contain both the model elements via the `model` property as well as the presentation elements via the `presentation` property. *PPMNModelPackages* are a specialization of *PartyModelPackage* and so may contain all of the *Party*-related elements contained therein.

ProfilePackages group elements associated with *AdornmentProfile* definitions so that profiles can be shared between organizations and/or user communities. These elements include *AdornmentProfiles*, *AdornmentDefinitions*, *AttributeTypes*, and *AdornmentAttributeDefinitions*.

All **PPMN** packages and models are specializations of *SCEPackage* and as such can contain other *SCEPackages* or their specializations. They can also include imports of external elements throught the *Import* element.

ProfilePackages group elements associated with *AdornmentProfile* definitions so that profiles can be shared between organizations and/or user communities. These elements include *AdornmentProfiles*, *AdornmentDefinitions*, *AttributeTypes*, and *AdornmentAttributeDefinitions*.

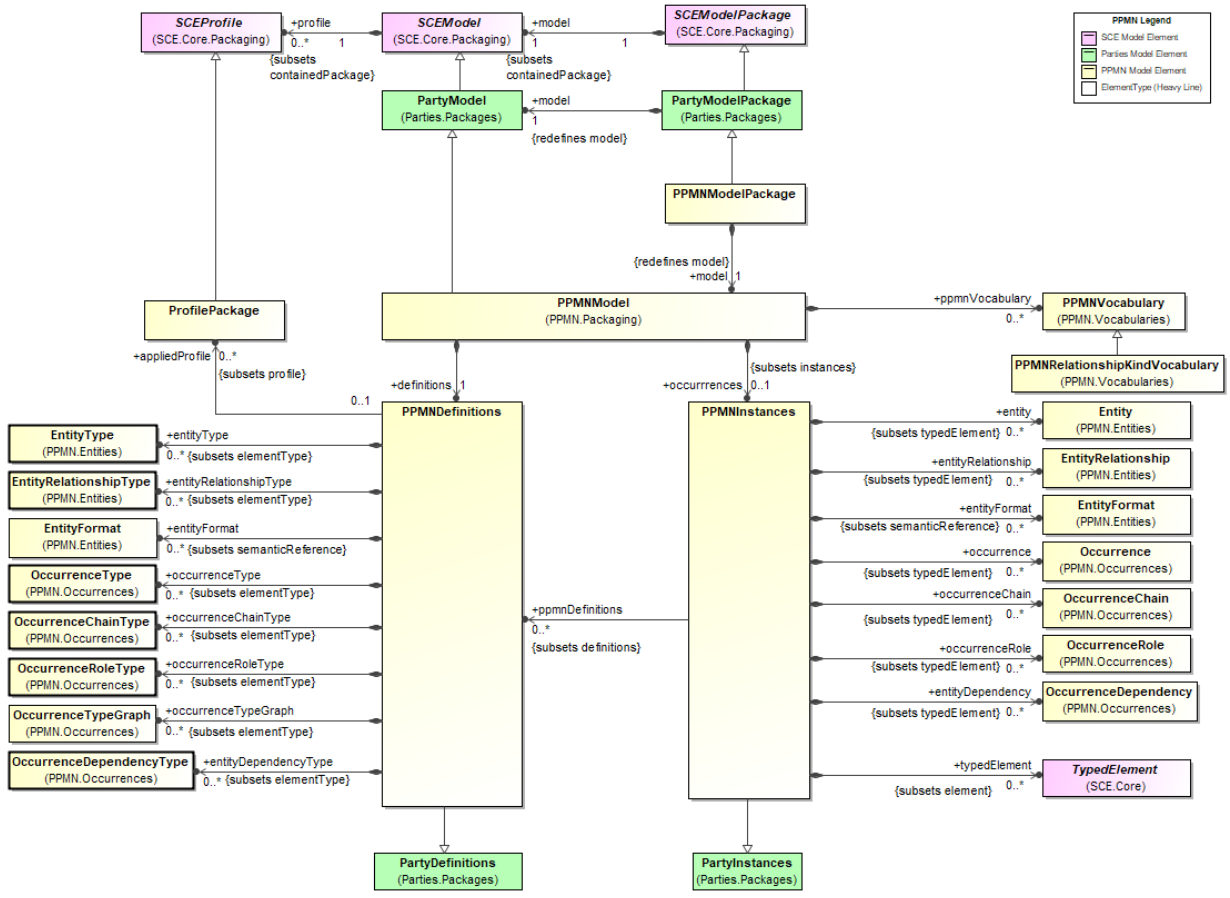


Figure 49: PPMN Packaging

8.10.1 PPMNDefinitions

A kind of *SCEDefinitions* that is the container for "Type-related" PPMN elements. Type-related elements include elements such as *OccurrenceChainTypes* and its specializations, *OccurrenceTypes* and its specializations, and

profiles. Type-related elements are contained in *TypePackages* while profiles are contained in *ProfilePackages*.

Generalizations

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

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

Properties

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

Table 82. PPMNDefinitions Attributes and/or Associations

Property/Association	Description
appliedProfile : ProfilePackage [0..*]	A set of *ProfilePackages* included in this package that contain any necessary profile definitions.
entityDependencyType : OccurrenceDependencyType [0..*]	A list of *EntityDependencyTypes* within the *PPMNModel*.
entityFormat : EntityFormat [0..*]	A list of the *EntityFormats* referenced within the *PPMNDefinitions* package.
entityRelationshipType : EntityRelationshipType [0..*]	A list of *EntityRelationshipTypes* within the *PPMNModel*.
entityType : EntityType [0..*]	A list of *EntitieTypes* within the *PPMNModel*.
occurrenceChainType : OccurrenceChainType [0..*]	A list of *OccurrenceChainTypes* within the *PPMNModel*.
occurrenceRoleType : OccurrenceRoleType [0..*]	A list of *OccurrenceRoleTypes* within the *PPMNModel*.
occurrenceType : OccurrenceType [0..*]	A list of *OccurrenceTypes* within the *PPMNModel*.
occurrenceTypeGraph : OccurrenceTypeGraph [0..*]	A list of *OccurrenceTypeGraphs* within the *PPMNModel*.

8.10.2 PPMNInstances

PPMN information sets are exchanged in bulk through the *OccurrenceSet* element. The *OccurrenceSet* element provides the outermost container for other **PPMN** elements contained in one or more *PPMNPackages*. The occurrence chains, occurrences and other "instance-related" elements are contained within one or more *OccurrenceSets* while "type-related" elements such as *OccurrenceChainTypes*, *OccurrenceTypes*, and *PPMNProfiles* if present are contained within *Definitions* packages.

Generalizations

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

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

Properties

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

Table 83. PPMNInstances Attributes and/or Associations

Property/Association	Description
entity : Entity [0..*]	A list of *Entities* of interest within the *PPMNModel*.
entityDependency : OccurrenceDependency [0..*]	A list of *EntityDependencies* within the *PPMNModel*.
entityFormat : EntityFormat [0..*]	A list of the *EntityFormats* referenced within the *PPMNInstances* package.
entityRelationship : EntityRelationship [0..*]	A list of *EntityRelationships* within the *PPMNModel*.
occurrence : Occurrence [0..*]	A list of *Occurrences* within the *PPMNModel*.
occurrenceChain : OccurrenceChain [0..*]	A list of *OccurrenceChains* within the *PPMNModel*.
occurrenceRole : OccurrenceRole [0..*]	A list of *OccurrenceRoles* within the *PPMNModel*.
ppmnDefinitions : PPMNDefinitions [0..*]	The property refers to zero or more *PPMNDefinitions* packages that contains the *ElementTypes* that provide a basis for the instances contained in the *PartyInstances* package.
typedElement : TypedElement [0..*]	A list of *TypedElements* within the *PPMNModel*.

8.10.3 PPMNModel

A *PPMNModel* is the main container for semantic elements of a **PPMN** model including types, instances, profiles, and vocabularies. As a specialization of *PartyModel* it also contains Party-related types, instances, profiles, and vocabularies. These elements are separate from the visual elements included in the PPMNModelPackage.

Generalizations

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

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

Properties

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

Table 84. PPMNModel Attributes and/or Associations

Property/Association	Description
definitions : PPMNDefinitions [1]	The packages that contain the elements that represent the definitions of a **PPMN** model. These elements generally include the types and profile elements.

occurrrences : PPMNInstances [0..1]	The packages that contain the elements that represent the definitions of a **PPMN** model. These elements generally include the types and profile elements.
ppmnVocabulary : PPMNVocabulary [0..*]	The `ppmnVocabulary` is a list of terms (as *SemanticReferences*) that provide an extensible mechanism to define the elements of enumerations in a *PPMNModel*.

8.10.4 PPMNModelPackage

A namespace that groups **PPMN** *Elements* comprising the pedigree and provenance information about some set of entities.

Generalizations

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

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

Properties

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

Table 85. PPMNModelPackage Attributes and/or Associations

Property/Association	Description
model : PPMNModel [1]	The *PPMNModel* contained within the *PPMNModelPackage*.

8.10.5 ProfilePackage

A kind of *PPMNPackage* that comprises **PPMN** profiles that can be applied to other **PPMN** *TypedElements*. *ProfilePackages* provide a mechanism to exchange profile libraries.

Generalizations

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

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

Properties

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

Table 86. ProfilePackage Attributes and/or Associations

Property/Association	Description
adornmentDefinition : AdornmentDefinition [0..*]	A set of *AdornmentDefinitions* contained within the package.
attributeType : AttributeType [0..*]	A set of *AttributeTypes* contained within the package.
profile : AdornmentProfile [0..*]	A set of *AdornmentProfiles* contained within the package.

profileAttributeDefinition : AdornmentAttributeDefinition [0..*]	A set of *AdornmentAttrributeDefinitions* contained within the package.

8.11 Primitives

The Primitives package contains primitive data elements used by other packages in PPMN.

PPMN uses the four primitives shown in the figure in addition to other UML primitives.

Figure 50: PPMN Primitives

8.11.1 DateTime

A primitive that captures a point in time including a date and the time of day to greatest precision practical.

Generalizations

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

Properties

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

8.12 Vocabularies

PPMNVocabularies are sets of terms used within a **PPMN** model that are defined by an external ontology. The terms link to formal definitions for the terms used within the model. The *SemanticReference* element, or a specialization thereof, is used to name the term and provide a link to the definitions. *PPMNVocabularies* are contained within a *PPMNModel* package.

The following figure presents the elements related to the *PPMNVocabulary* section:

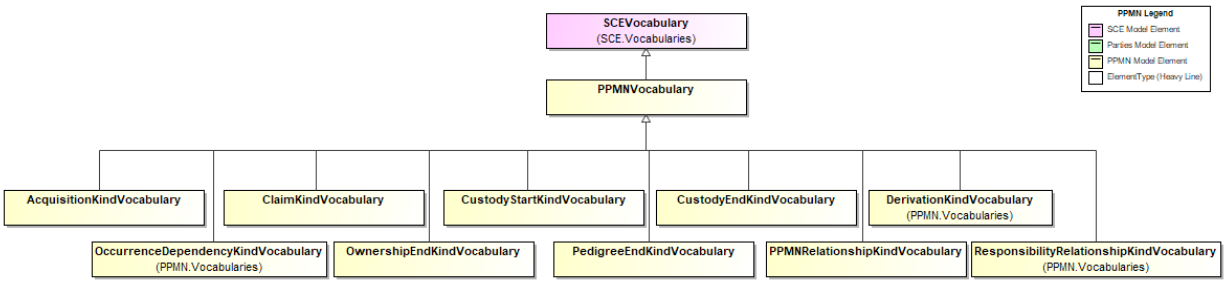


Figure 51: PPMNVocabulary

8.12.1 PPMNVocabulary

PPMNVocabularies are lists of terms used as possible values for properties within PPMN. The terms are specializations of the *SemanticReference* element that can be used to relate to the term to an external definition or meaning. The terms themselves do not represent the definitions or meanings but provide links to an external source. The vocabulary mechanism is used to support extensibility of the specification.

Generalizations

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

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

Properties

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

8.12.2 AcquisitionKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify how a *ChainOfOwnership* was started.

Generalizations

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

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

Properties

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

Table 87. AcquisitionKindVocabulary Attributes and/or Associations

Property/Association	Description
term : AcquisitionKind [1..*]	A list of the terms representing valid *AcquisitionKinds*.

8.12.3 ClaimKindVocabulary

A kind of *PPMNVocabulary* that includes terms that indicate the kind of *Claim* that has been made.

Generalizations

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

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

Properties

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

Table 88. ClaimKindVocabulary Attributes and/or Associations

Property/Association	Description
term : ClaimKind [1..*]	A list of the terms representing valid *ClaimKinds* within a **PPMN** Model.

8.12.4 CustodyEndKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify how a *ChainOfCustody* was ended.

Generalizations

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

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

Properties

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

Table 89. CustodyEndKindVocabulary Attributes and/or Associations

Property/Association	Description
term : CustodyEndKind [1..*]	A list of the terms representing valid *CustodyEndKinds* within a **PPMN** Model.

8.12.5 CustodyStartKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify how a *ChainOfCustody* was started.

Generalizations

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

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

Properties

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

Table 90. CustodyStartKindVocabulary Attributes and/or Associations

Property/Association	Description
term : CustodyStartKind [1..*]	A list of the terms representing valid *CustodyStartKinds* within a **PPMN** Model.

8.12.6 DerivationKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify the type of derivation relationship that exists between two *Entities*.

Generalizations

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

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

Properties

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

Table 91. DerivationKindVocabulary Attributes and/or Associations

Property/Association	Description
term : DerivationKind [1..*]	A list of the terms representing valid *DerivationTypes* within a **PPMN** Model.

8.12.7 OccurrenceDependencyKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify how the type of dependency an *Occurrence* has on an *Entity*.

Generalizations

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

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

Properties

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

Table 92. OccurrenceDependencyKindVocabulary Attributes and/or Associations

Property/Association	Description
term : OccurrenceDependencyKind [1..*]	A list of the terms representing valid OccurrenceDependencies within a **PPMN** Model.

8.12.8 OwnershipEndKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify how the *ChainOfOwnership* was ended.

Generalizations

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

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

Properties

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

Table 93. OwnershipEndKindVocabulary Attributes and/or Associations

Property/Association	Description
term : OwnershipEndKind [1..*]	A list of the terms representing valid *OwnershipEndKinds* within a **PPMN** Model.

8.12.9 PedigreeEndKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify the kind of relationship between two **PPMN** elements.

Generalizations

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

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

Properties

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

8.12.10 PPMNRelationshipKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify the kind of relationship between two **PPMN** elements.

Generalizations

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

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

Properties

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

Table 94. PPMNRelationshipKindVocabulary Attributes and/or Associations

Property/Association	Description
term : RelationshipKind [0..*]	A list of the terms representing valid *RelationshipKinds* within a **PPMN** model.

8.12.11 ResponsibilityRelationshipKindVocabulary

A kind of *PPMNVocabulary* that includes terms that specify the kind of *ResponsibilityRelationship* exists between one or more *Parties* and an *Entity*.

Generalizations

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

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

Properties

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

Table 95. ResponsibilityRelationshipKindVocabulary Attributes and/or Associations

Property/Association	Description
term : ResponsibilityRelationshipKind [0..*]	A list of the terms representing valid *RelationshipKinds* within a **PPMN** model.

9 PPMN Library

A Library is included in **PPMN** to provide standard instances that are intended to be implemented by tools supporting **PPMN**. Currently, **PPMN** defines the instances for *AcquisitionKinds*, *ClaimKinds*, *CustodyStartKinds*, *CustodyEndKinds*, *OwnershipEndKinds*, *PedigreeEndKinds*, and *RelationshipKinds* (See following sections).

9.1 AcquisitionKinds

The *AcquisitionKinds* library contains instances that represent the standard ways in which ownership of an entity may begin. These elements are instances of *AcquisitionKind*. The vocabulary can be extended with additional instances of *AcquisitionKind* or a specialization thereof.

The following figure presents the instances of the *AcquisitionKind* element that are `terms` for the `AcquisitionKindsVocabulary`:

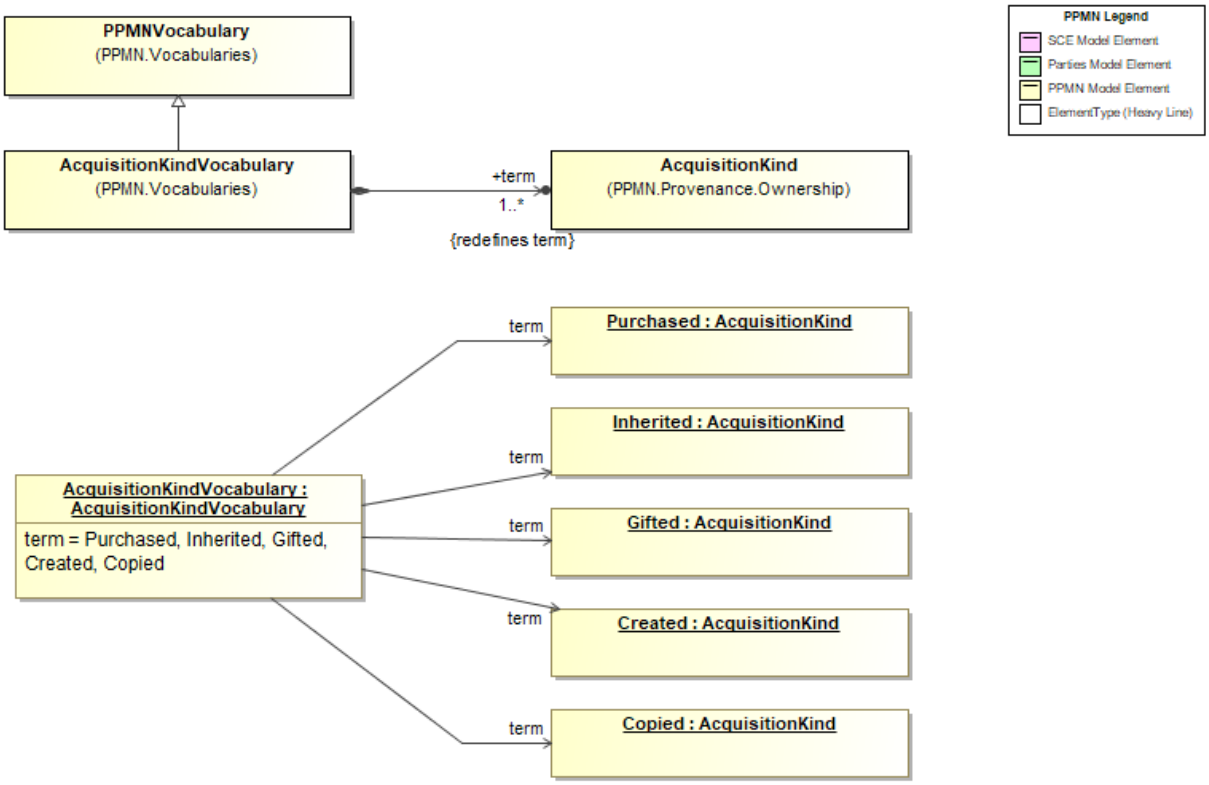


Figure 52: AcquisitionKinds

The following table provides a definition of the terms included in the *AcquisitionKinds* Vocabulary.

Table 96. AcquisitionKinds Vocabulary

#	Name	Documentation
1	AcquisitionKindVocabulary	A kind of PPMNVocabulary that includes terms that specify how a *ChainOfOwnership* was started.
2	Copied	An instance that indicates that a Party gained ownership of an entity by copying another entity.
3	Created	An instance that indicates that a Party gained ownership of an entity by creating it.
4	Gifted	An instance that indicates that a Party gained ownership of an entity by receiving it as a gift.
5	Inherited	An instance that indicates that a Party gained ownership of an entity as part of an inheritance.
6	Purchased	An instance that indicates that a Party gained ownership of an entity by purchasing the entity.

9.1.1 AcquisitionKindVocabulary

A kind of PPMNVocabulary that includes terms that specify how a *ChainOfOwnership* was started.

9.1.2 Copied

An instance that indicates that a Party gained ownership of an entity by copying another entity.

9.1.3 Created

An instance that indicates that a Party gained ownership of an entity by creating it.

9.1.4 Gifted

An instance that indicates that a Party gained ownership of an entity by receiving it as a gift.

9.1.5 Inherited

An instance that indicates that a Party gained ownership of an entity as part of an inheritance.

9.1.6 Purchased

An instance that indicates that a Party gained ownership of an entity by purchasing the entity.

9.2 ClaimKinds

The *ClaimKinds* library contains instances that represent the standard types of claims that can be made in regards to a set of PPMN elements. These elements are instances of *ClaimKind*. The vocabulary can be extended with additional instances of *ClaimKinds* or a specialization thereof.

The following figure presents the instances of the *ClaimKind* element that are `terms` for the `ClaimKindsVocabulary`:

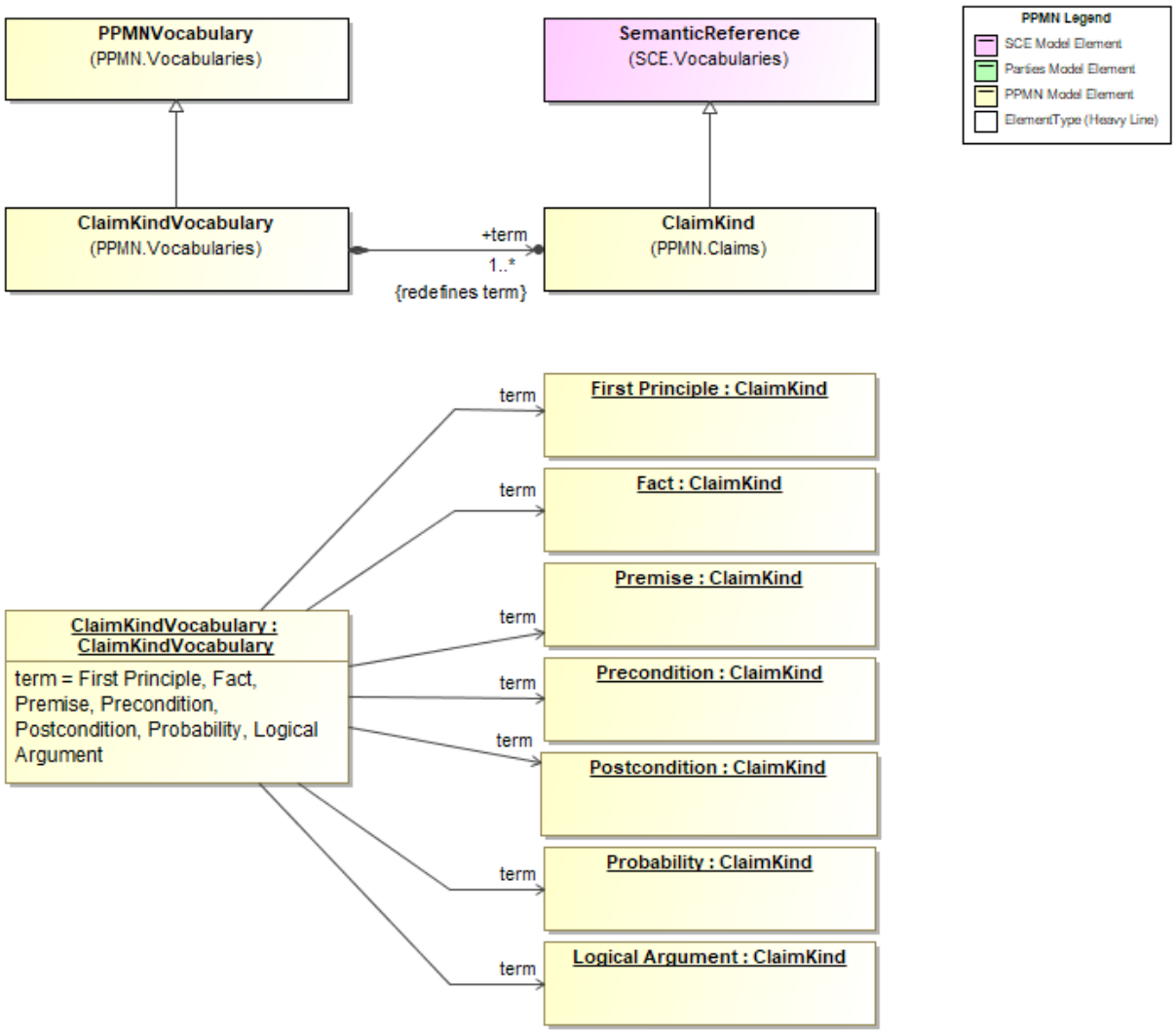


Figure 53: ClaimKinds

The following table provides a definition of the terms included in the *ClaimKinds* Vocabulary.

Table 97. ClaimKinds Vocabulary

#	Name	Documentation
1	ClaimKindVocabulary	An vocabulary of terms that specify the kinds of claims may be made.
2	Fact	A basic assertion.
3	First Principle	A foundational assertion that is held as true.
4	Logical Argument	An assertion that is based on other assertions.
5	Postcondition	An assertion that is assumed to be true at the end of a process.

#	Name	Documentation
6	Precondition	An assertion that is assumed to be true at the start of a process.
7	Premise	An assertion that is used in a logical argument.
8	Probability	An assertion that indicates some degree of truth.

9.2.1 ClaimKindVocabulary

An vocabulary of terms that specify the kinds of claims may be made.

9.2.2 Fact

A basic assertion.

9.2.3 First Principle

A foundational assertion that is held as true.

9.2.4 Logical Argument

An assertion that is based on other assertions.

9.2.5 Postcondition

An assertion that is assumed to be true at the end of a process.

9.2.6 Precondition

An assertion that is assumed to be true at the start of a process.

9.2.7 Premise

An assertion that is used in a logical argument.

9.2.8 Probability

An assertion that indicates some degree of truth.

9.3 CustodyEndKinds

The *CustodyEndKinds* library contains instances that represent the standard ways in which custody of an entity may end. These elements are instances of *CustodyEndKind*. The vocabulary can be extended with additional instances of *CustodyEndKind* or a specialization thereof.

The following figure presents the instances of the *CustodyEndKind* element that are `terms` for the `CustodyEndKindsVocabulary`:

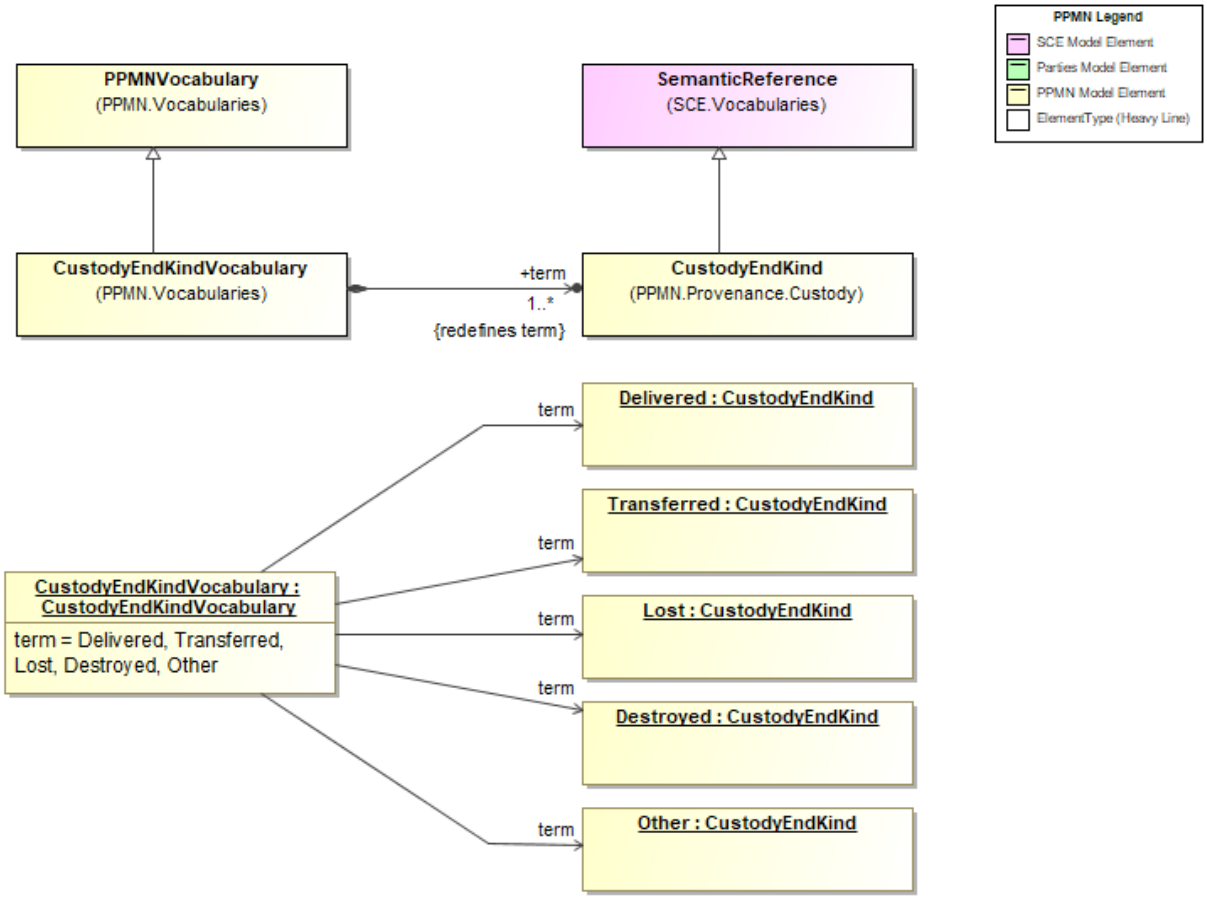


Figure 54: CustodyEndKinds

The following table provides a definition of the terms included in the *CustodyEndKinds* Vocabulary.

Table 98. CustodyEndKinds Vocabulary

#	Name	Documentation
1	CustodyEndKindVocabulary	An vocabulary of terms that specify the kind of *CustodyOccurrence* that results in the end of a *ChainOfCustody*.
2	Delivered	An instance that specifies that an entity was delivered to some other *Party*.
3	Destroyed	An instance that specifies that an entity was destroyed.
4	Lost	An instance that specifies that an entity was lost.
5	Other	An instance that specifies that custody of an entity was relinquished in some other way.

#	Name	Documentation
6	Transferred	An instance that specifies that an entity was transferred to some other *Party*.

9.3.1 CustodyEndKindVocabulary

A vocabulary of terms that specify the kind of *CustodyOccurrence* that results in the end of a *ChainOfCustody*.

9.3.2 Delivered

An instance that specifies that an entity was delivered to some other *Party*.

9.3.3 Destroyed

An instance that specifies that an entity was destroyed.

9.3.4 Lost

An instance that specifies that an entity was lost.

9.3.5 Other

An instance that specifies that custody of an entity was relinquished in some other way.

9.3.6 Transferred

An instance that specifies that an entity was transferred to some other *Party*.

9.4 CustodyStartKinds

The *CustodyStartKinds* library contains instances that represent the standard ways in which custody of an entity may begin. These elements are instances of *CustodyStartKind*. The vocabulary can be extended with additional instances of *CustodyStartKind* or a specialization thereof.

The following figure presents the instances of the *CustodyStartKind* element that are `terms` for the `CustodyStartKindsVocabulary`:

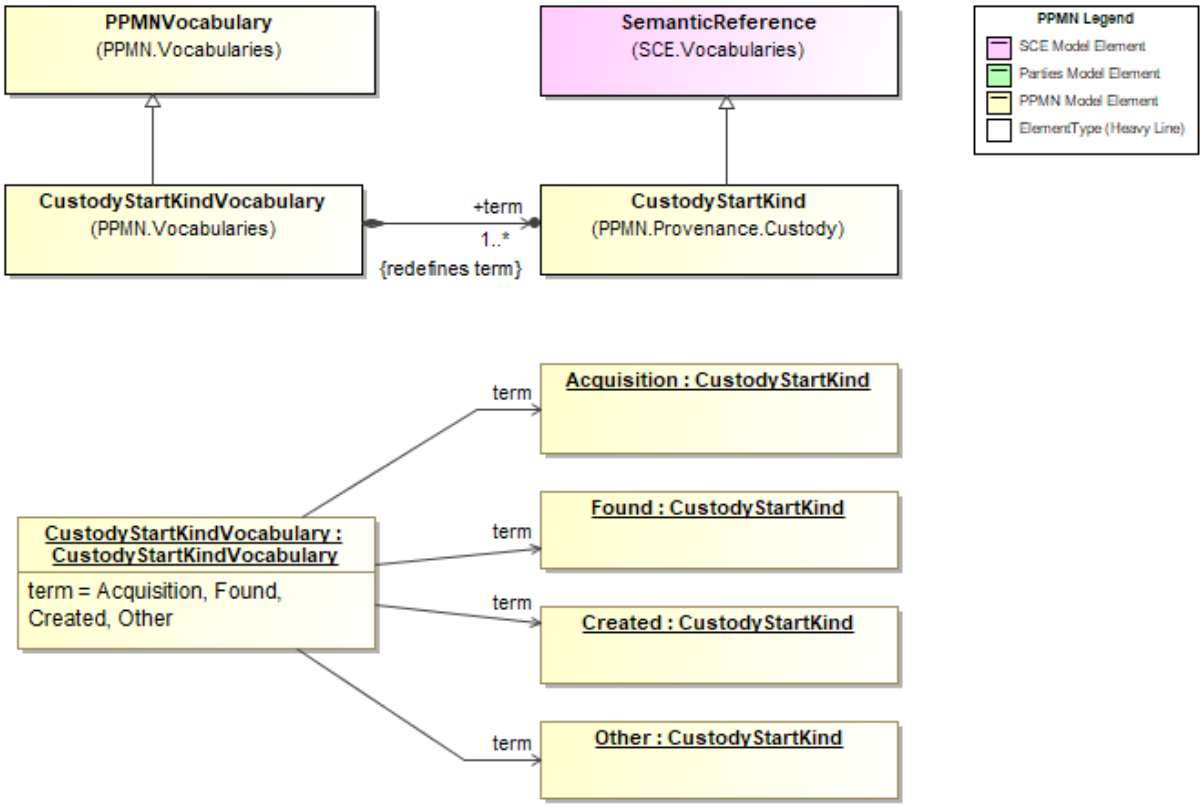


Figure 55: CustodyStartKinds

The following table provides a definition of the terms included in the *CustodyStartKinds* Vocabulary.

Table 99. CustodyStartKinds Vocabulary

#	Name	Documentation
1	CustodyStartKindVocabulary	A vocabulary of terms that specify the kind of *CustodyOccurrence* that results in the start of a *ChainOfCustody*.
2	Acquisition	An instance that indicates that a *Party* gains custody of an entity through some type of acquisition.
3	Created	An instance that indicates that a *Party* gains custody of an entity by creation of the entity.
4	Found	An instance that indicates that a *Party* gains custody of an entity when the entity is found.
5	Other	An instance that indicates that a *Party* gains custody of an entity by some other event.

9.4.1 CustodyStartKindVocabulary

A vocabulary of terms that specify the kind of *CustodyOccurrence* that results in the start of a *ChainOfCustody*.

9.4.2 Acquisition

An instance that indicates that a *Party* gains custody of an entity through some type of acquisition.

9.4.3 Created

An instance that indicates that a *Party* gains custody of an entity by creation of the entity.

9.4.4 Found

An instance that indicates that a *Party* gains custody of an entity when the entity is found.

9.4.5 Other

An instance that indicates that a *Party* gains custody of an entity by some other event.

9.5 DerivationKinds

The following table provides a definition of the terms included in the *DerivationKinds* Vocabulary.

Table 100. DerivationtKinds Vocabulary

#	Name	Documentation
1	DerivationKindsVocabulary	A vocabulary of terms that specify the kind of derivation that exists between two *Entities*.
2	DerivedFrom	`DerivedFrom` indicates that `source` *EntityTypes* are derived in some way from `target` *EntityTypes*.
3	QuotedFrom	`QuotedFrom` indicates that `source` *EntityTypes* are quoted from `target` *EntityTypes*.
4	RevisionOf	`RevisionOf` indicates that `source` *EntityTypes* are revisions of `target` *EntityTypes*.
5	SourcedFrom	`SouredFrom` indicates that `source` *EntityTypes* are sourced from from `target` *EntityTypes* which in turn are produced by some party potentially with some special experience or knowledge.

The following figure presents the instances of the *RelationshipKind* element that are `terms` for the `PPMNRelationshipKindsVocabulary`:

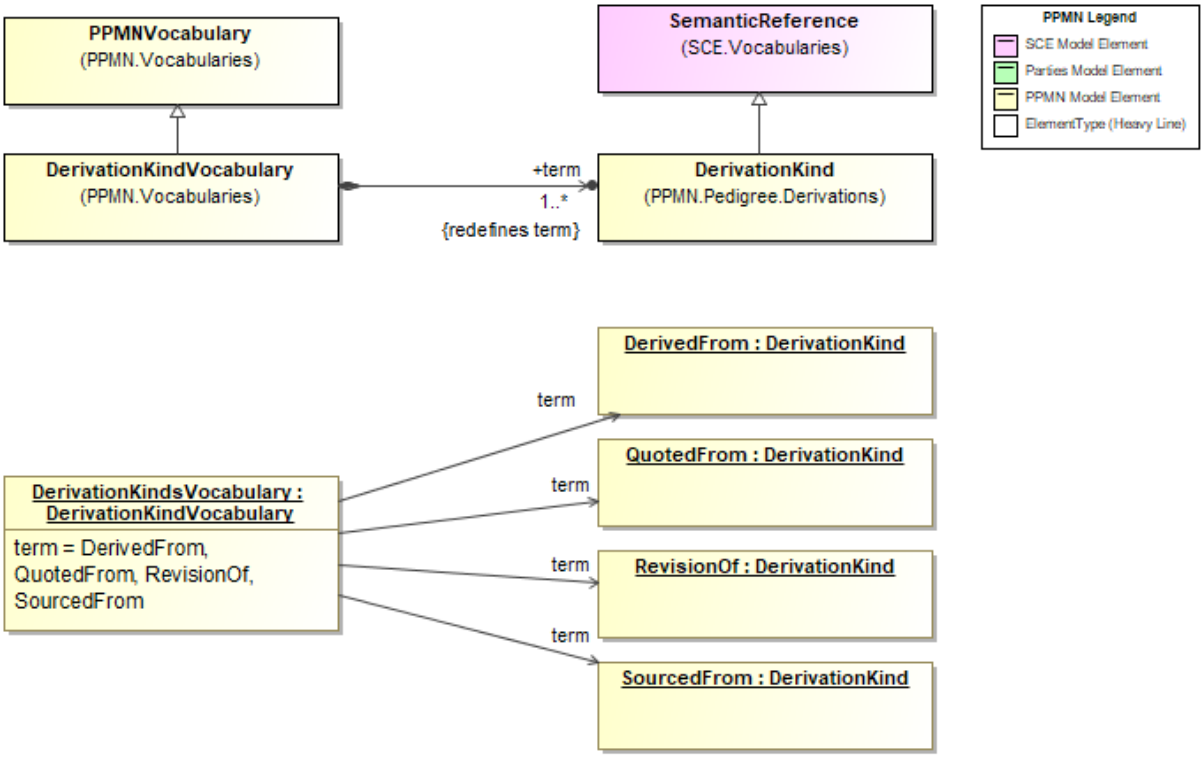


Figure 56: DerivationKinds

9.5.1 DerivationKindsVocabulary

A vocabulary of terms that specify the kind of derivation that exists between two *Entities*.

9.5.2 DerivedFrom

`DerivedFrom` indicates that `source` *EntityTypes* are derived in some way from `target` *EntityTypes*.

9.5.3 QuotedFrom

`QuotedFrom` indicates that `source` *EntityTypes* are quoted from `target` *EntityTypes*.

9.5.4 RevisionOf

`RevisionOf` indicates that `source` *EntityTypes* are revisions of `target` *EntityTypes*.

9.5.5 SourcedFrom

`SouredFrom` indicates that `source` *EntityTypes* are sourced from from `target` *EntityTypes* which in turn are produced by some party potentially with some special experience or knowledge.

9.6 OccurrenceDependencyKinds

The *OccurrenceDependencyKinds* library contains instances that represent the standard ways in which an *Occurrence* may depend on an *Entity*. These elements are instances of *OccurrenceDependencyKind*. The vocabulary can be extended with additional instances of *OccurrenceDependencyKind* or a specialization thereof.

The following figure presents the instances of the *OccurrencDependencyKind* element that are `terms` for the `OccurrenceDependencyKindsVocabulary`:

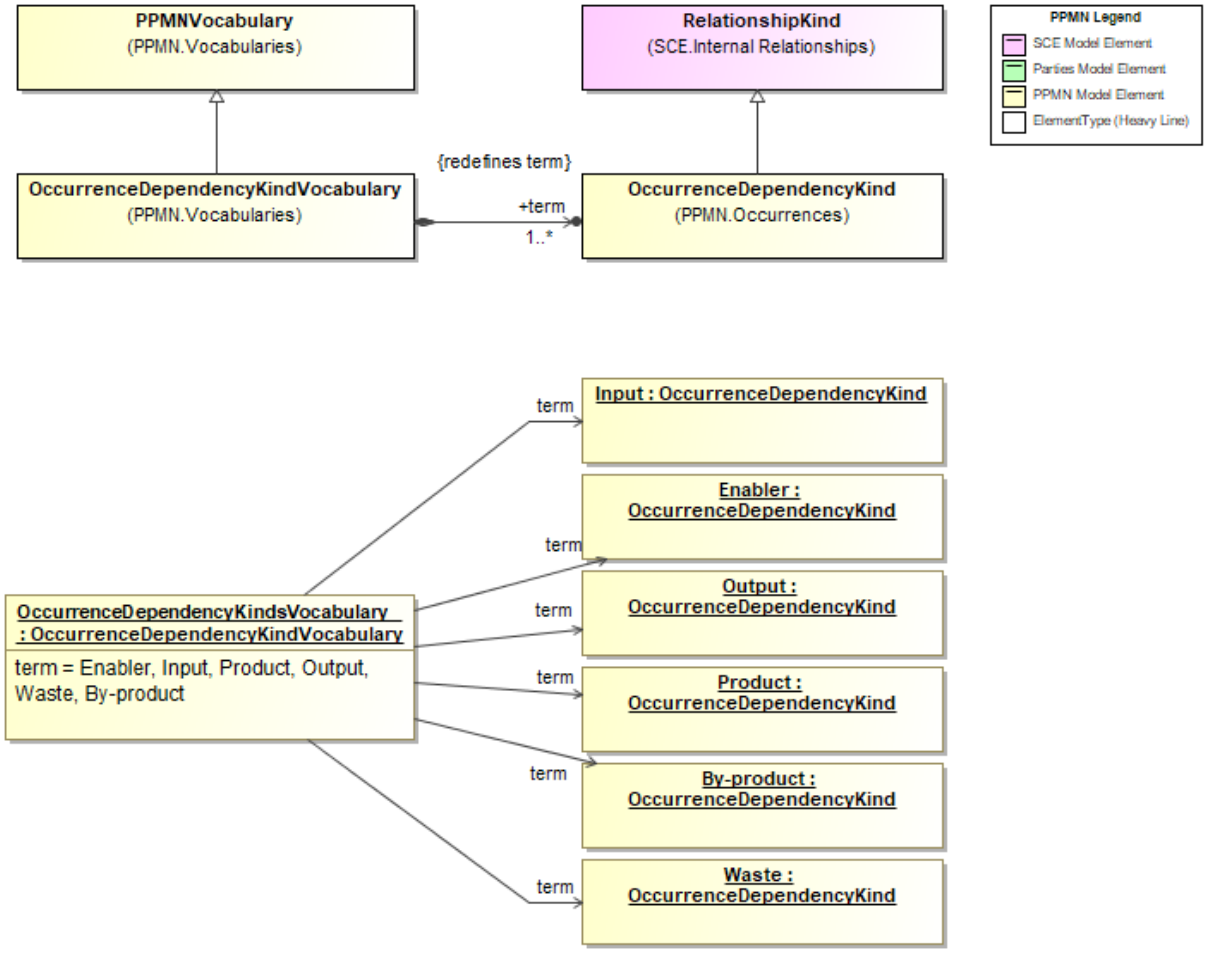


Figure 57: OccurrenceDependencyKinds

The following table provides a definition of the terms included in the *OccurrenceDependencyKinds* Vocabulary.

Table 101. OccurrenceDependencyKinds Vocabulary

#	Name	Documentation
1	OccurrenceDependencyKindsVocabulary	
2	By-product	`By-product` indicates that the `source` *Occurrence* produces or creates the `target` *Entity* as a by-product during the course of the *Occurrence*.

#	Name	Documentation
3	Enabler	`Enabler` indicates that the `source` *Occurrence* uses the `target` *Entity* in some way that enables the *Occurrence.* However, the *Entity* is not used or become a part of any of the products or by-products of the *Occurrence.*
4	Input	`Input` indicates that the `target` *Entity* is an input to the `source` *Occurrence* is an input during the course of the *Occurrence*.
5	Output	`Output` indicates that the `target` *Entity* is an output of some kind of the *Occurrence.*.
6	Product	`Product` indicates that the `source` *Occurrence* produces or creates the `target` *Entity* during the course of the *Occurrence*.
7	Waste	`Waste` indicates that the `source` *Occurrence* produces or creates the `target` *Entity* as waste during the course of the *Occurrence*.

9.6.1 OccurrenceDependencyKindsVocabulary

9.6.2 By-product

`By-product` indicates that the `source` *Occurrence* produces or creates the `target` *Entity* as a by-product during the course of the *Occurrence*.

9.6.3 Enabler

`Enabler` indicates that the `source` *Occurrence* uses the `target` *Entity* in some way that enables the *Occurrence.* However, the *Entity* is not used or become a part of any of the products or by-products of the *Occurrence.*

9.6.4 Input

`Input` indicates that the `target` *Entity* is an input to the `source` *Occurrence* is an input during the course of the *Occurrence*.

9.6.5 Output

`Output` indicates that the `target` *Entity* is an output of some kind of the *Occurrence.*.

9.6.6 Product

`Product` indicates that the `source` *Occurrence* produces or creates the `target` *Entity* during the course of the *Occurrence*.

9.6.7 Waste

`Waste` indicates that the `source` *Occurrence* produces or creates the `target` *Entity* as waste during the course of the *Occurrence*.

9.7 OwnershipEndKinds

The *OwnershipEndKinds* library contains instances that represent the standard ways in which ownership of an entity may end. These elements are instances of *OwnershipEndKind*. The vocabulary can be extended with additional instances of *OwnershipEndKind* or a specialization thereof.

The following figure presents the instances of the *OwnershipEndKind* element that are `terms` for the `OwnershipEndKindsVocabulary`:

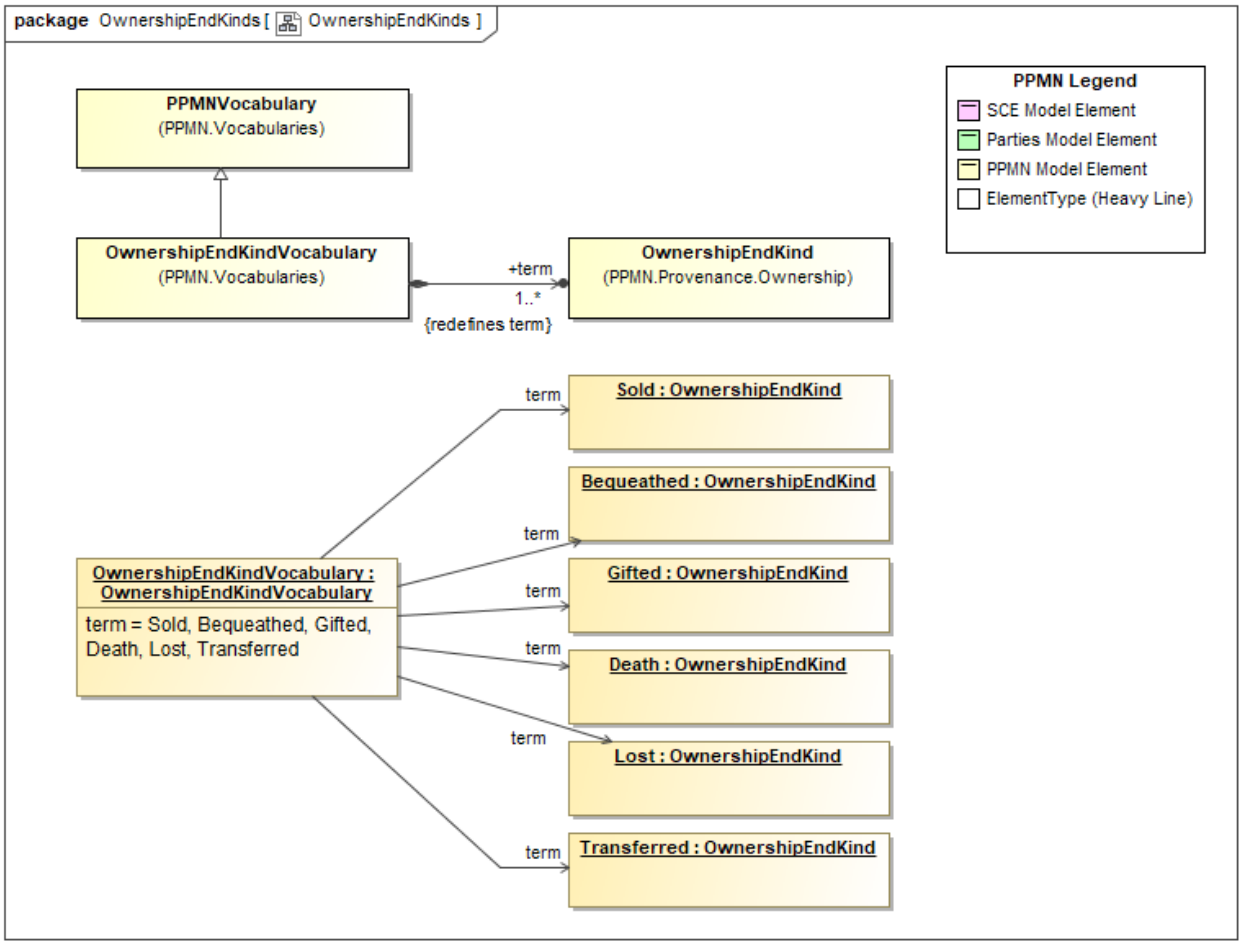


Figure 58: OwnershipEndKinds

The following table provides a definition of the terms included in the *OwnershipEndKinds* Vocabulary.

Table 102. OwnershipEndKinds Vocabulary

#	Name	Documentation
1	OwnershipEndKindVocabulary	A vocabulary of terms that specify how the *ChainOfOwnership* was ended.
2	Bequeathed	An instance that specifies that an entity was bequeathed to some other party.
3	Death	An instance that specifies that an entity died.
4	Gifted	An instance that specifies that an entity was gifted to some other *Party*.
5	Lost	An instance that specifies that an entity was lost.
6	Sold	An instance that specifies that an entity was sold to some other *Party*.
7	Transferred	An instance that specifies that ownership of an entity was transferred to some other *Party*.

9.7.1 OwnershipEndKindVocabulary

A vocabulary of terms that specify how the *ChainOfOwnership* was ended.

9.7.2 Bequeathed

An instance that specifies that an entity was bequeathed to some other party.

9.7.3 Death

An instance that specifies that an entity died.

9.7.4 Gifted

An instance that specifies that an entity was gifted to some other *Party*.

9.7.5 Lost

An instance that specifies that an entity was lost.

9.7.6 Sold

An instance that specifies that an entity was sold to some other *Party*.

9.7.7 Transferred

An instance that specifies that ownership of an entity was transferred to some other *Party*.

9.8 PPMNRelationshipKinds

The *PPMNRelationshiipKinds* library contains instances that represent the standard types of relationships between **PPMN** elements. This library extends the **SCE** RelationshipKinds library of terms to add `Transition`. These

elements are instances of **SCE** *RelationshiipKind*. The vocabulary can be extended with additional instances of *RelationshiipKind* or a specialization thereof.

The following figure presents the instances of the *RelationshipKind* element that are `terms` for the `PPMNRelationshipKindsVocabulary`:

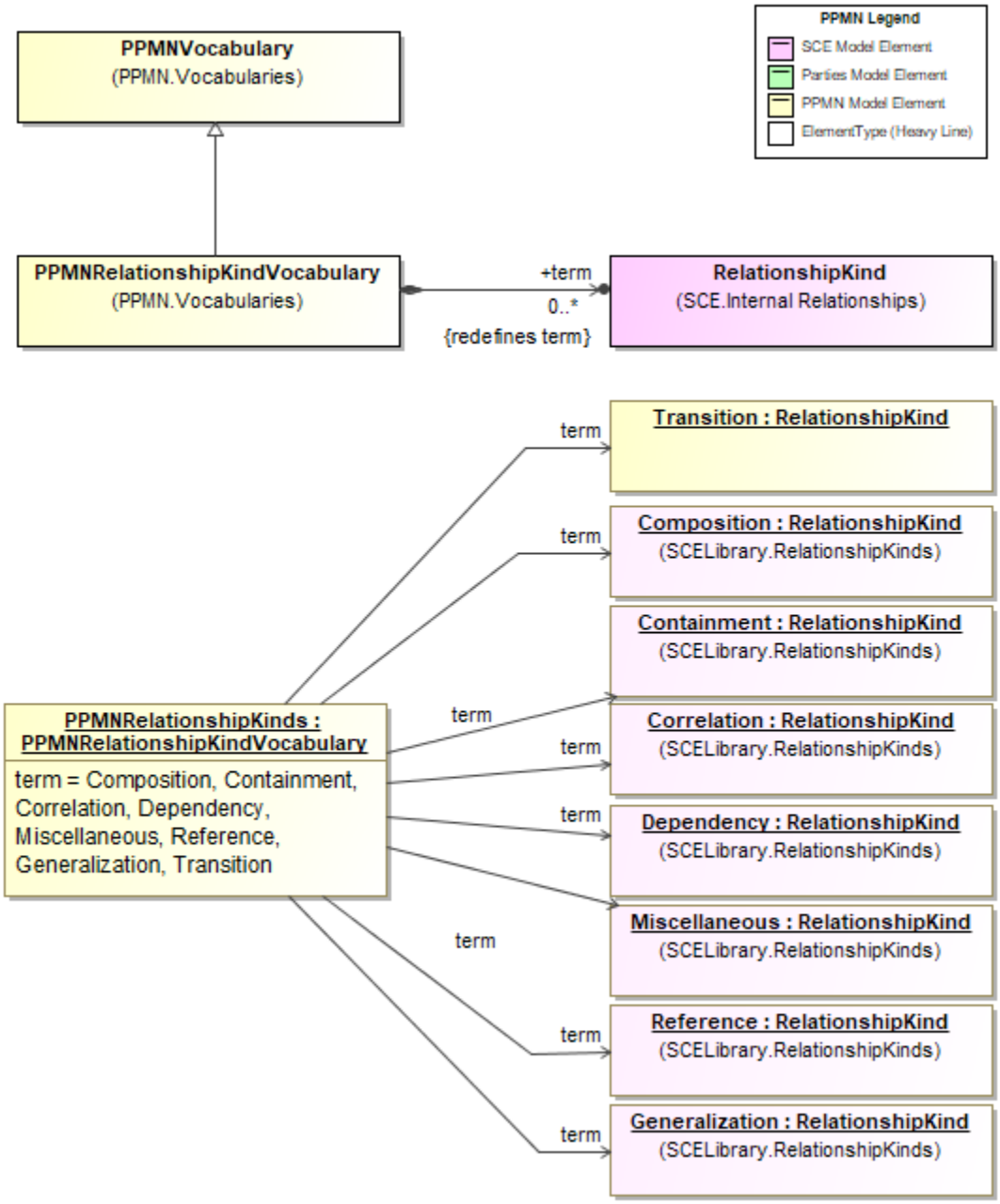


Figure 59: PPMNRelationshipKinds

The following table provides a definition of the terms included in the *PPMNRelationshipKinds* Vocabulary.

Table 103. PPMNRelationshipKinds Vocabulary

#	Name	Documentation
1	PPMNRelationshipKinds	A kind of PPMNVocabulary that includes terms that specify the kind of relationship between two **PPMN** elements.
2	Composition	`Composition` indicates that the `source` element is composed of, in part, the `target` element. Other elements could be included in this composition.
3	Containment	`Containment` indicates that the `source` element is a container for the `target` element.
4	Correlation	`Correlation` indicates that the `source` element is correlated with the `target` element. This is often used when a mapping is required between the structures of two data elements.
5	Dependency	`Dependency` indicates that `target` element is dependent in some way on the `source` element.
6	Generalization	`Generalization` indicates that the `source` element is a generalization of the `target` element (which is based on and extends the `source`).
7	Miscellaneous	`Miscellaneous` indicates that `source` element has some relationship with the `target` element that is of a kind that is not expressed through the other *RelationshipKind* instances.
8	Reference	`Reference` indicates that `source` element references the `target` element.
9	Transition	`Transition` indicates that "flow" or sequencing moves from the `source` element to the `target` element.

9.8.1 PPMNRelationshipKinds

A kind of PPMNVocabulary that includes terms that specify the kind of relationship between two **PPMN** elements.

9.8.2 Transition

`Transition` indicates that "flow" or sequencing moves from the `source` element to the `target` element.

9.8.3 Additional Terms from SCE

9.8.3.1 Reference

`Reference` indicates that `source` element references the `target` element.

9.8.3.2 Miscellaneous

`Miscellaneous` indicates that `source` element has some relationship with the `target` element that is of a kind that is not expressed through the other *RelationshipKind* instances.

9.8.3.3 Composition

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

9.8.3.4 Dependency

`Dependency` indicates that `target` element is dependent in some way on the `source` element.

9.8.3.5 Containment

`Containment` indicates that the `source` element is a container for the `target` element.

9.8.3.6 Correlation

`Correlation` indicates that the `source` element is correlated with the `target` element. This is often used when a mapping is required between the structures of two data elements.

9.8.3.7 Generalization

`Generalization` indicates that the `source` element is a generalization of the `target` element (which is based on and extends the `source`).

9.9 ResponsibilityRelationshipKinds

The following figure presents the instances of the *ResponsibilityRelationshipKind* element that are `terms` for the `ResponsibilityRelationshipKindsVocabulary`:

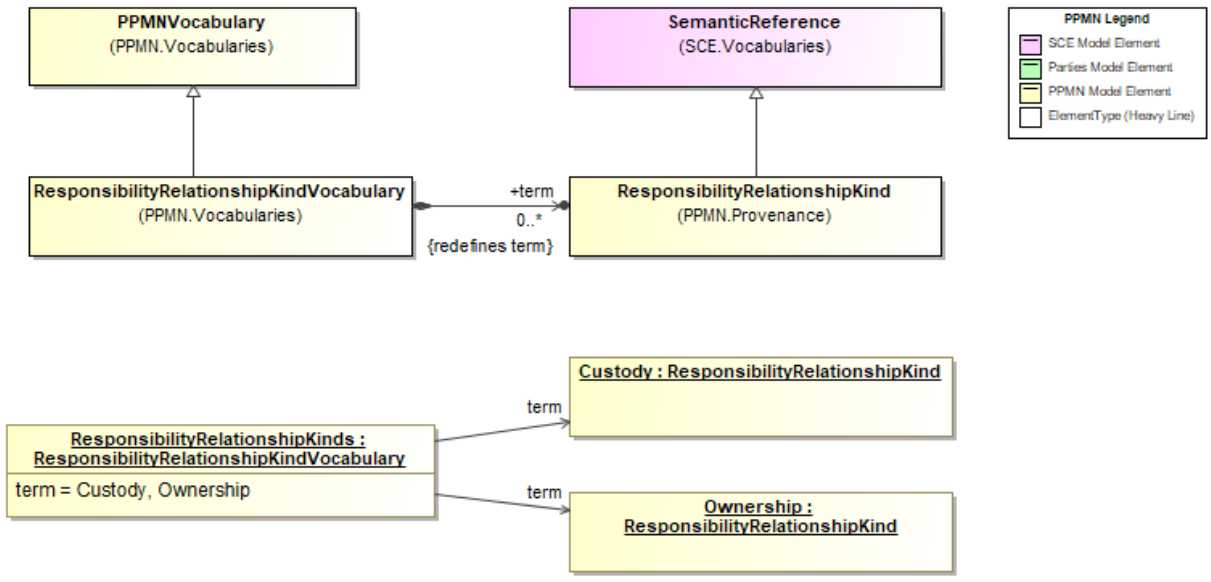


Figure 60: ResponsibilityRelationshipKinds

The following table provides a definition of the terms included in the *PPMNRelationshipKinds* Vocabulary.

Table 104. ResponsibilityRelationshipKinds Vocabulary

#	Name	Documentation
1	ResponsibilityRelationshipKinds	A kind of PPMNVocabulary that includes terms that specify the kind of responsibility a *Party* has with respect to an *Entity*.
2	Custody	`Custody` indicates that the `source` element has custody of the `target` element.
3	Ownership	`Ownership` indicates that the `source` element owns the `target` element.

9.9.1 ResponsibilityRelationshipKinds

A kind of PPMNVocabulary that includes terms that specify the kind of responsibility a *Party* has with respect to an *Entity*.

9.9.2 Custody

`Custody` indicates that the `source` element has custody of the `target` element.

9.9.3 Ownership

`Ownership` indicates that the `source` element owns the `target` element.

10 Parties Model

This section defines the semantic elements of the **Parties** Metamodel. The main topics are organized into Core Elements, Locations, Packages, Vocabularies, and Primitives.

10.1 Core

The Core elements of the **Parties** metamodel contains elements related to people, organizations, roles, automated systems and the relationships between them. The elements are separated into Instances and Types. The Instances section defines elements that enable modeling specific Parties (i.e., people, organizations, positions and roles and their interrelationships). The Types section defines elements that enable modeling the kinds of Parties that are of interest in some context.

10.1.1 Instances

The Core.Intances section of the **Parties** metamodel contains elements related to people, organizations, roles, automated systems and the relationships between them. These elements enable modeling specific Parties. Elements in the Core.Instances section are generally specializations of **SCE** *TypedElements* and as such may have an ElementType specified. The corresponding types are described below in the Core.Types section.

A *Party* is an abstract concept intended to generalize the notions of *Organization*, *Person*, *Position* or *Non-Human Agent* - essentially things that can be proactive and play a part in a business context. This generalization acknowledges the fact that many of the same business interactions can be defined regardless of the particular type of party involved. For instance, in the sale of a parcel of land, the seller might be a *Person* or an *Organization* or even a *Position* in an *Organization* wherein that *Position* is responsible for handling real estate transactions. Likewise for the buyer. The *Party* pattern captures this notion in a succinct manner that has broad applicability.

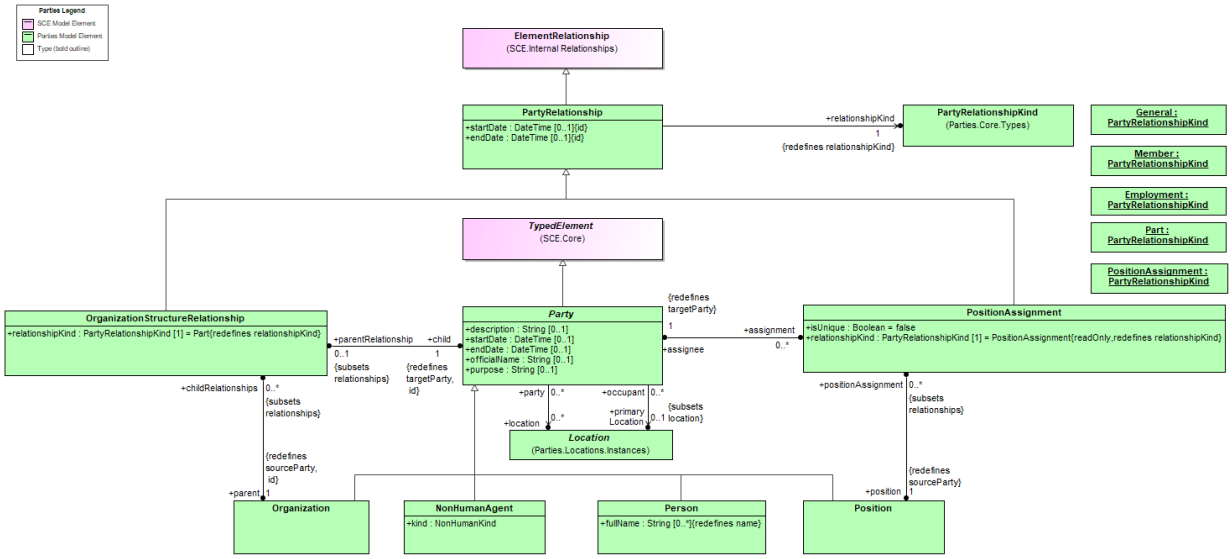


Figure 61: Parties

PartyRelationships capture relationships between *Parties.* The precise kind of relationship is specified by the `relationshipKind` property. There are two specializations of PartyRelationship: *OrganizationalStructureRelationship* and *PositionAssignment*. *OrganizationalStructureRelationship* supports the specification of the structure of an *Organization* while *PositionAssignment* supports the assignment of *Parties* to *Positions.*

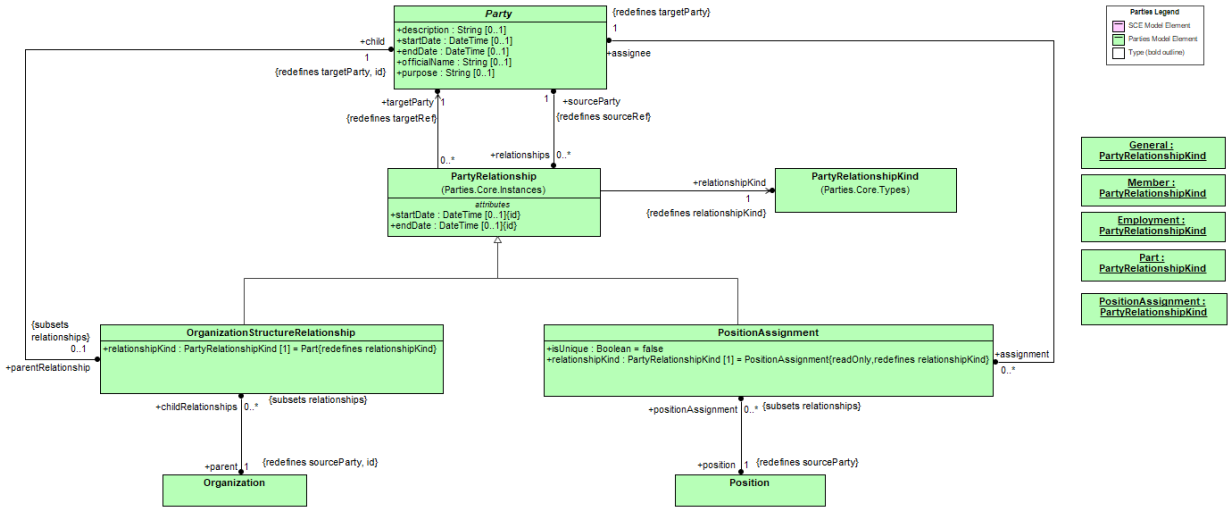


Figure 62: Party Relationships

Delegation captures the notion that a Party may assign a set of responsibilities to another party. The responsibilities being assigned are essentially captured as a *Position*.

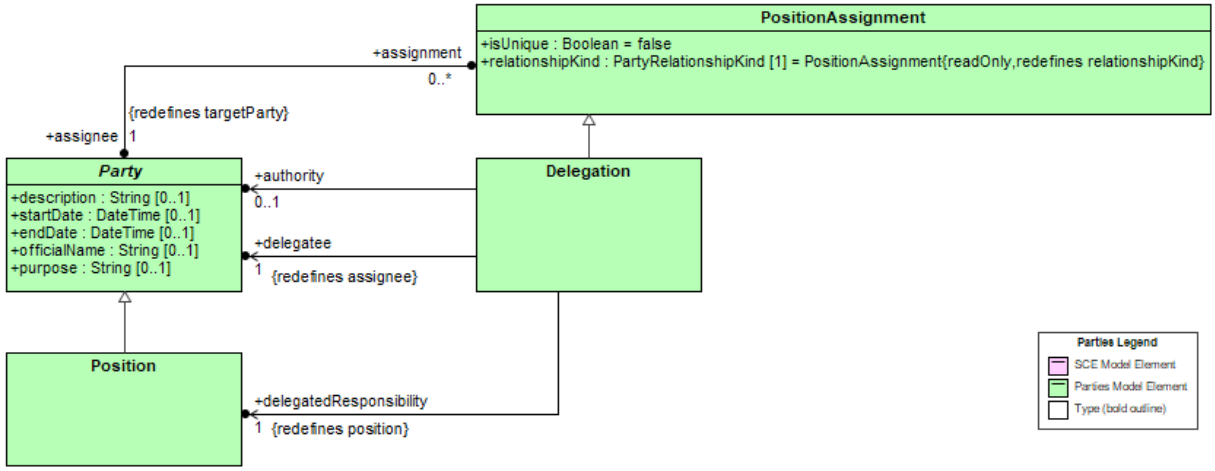


Figure 63: Delegation

PartyRoles represent a role that a *Party* may play in some context. For instance, in the sale of a parcel of land, the Seller might be a *Person* or an *Organization* or even a *Position* in an *Organization* wherein that *Position* is responsible for handling real estate transactions. Likewise for the buyer. The *PartyRole* captures this notion in a succinct manner that has broad applicability.

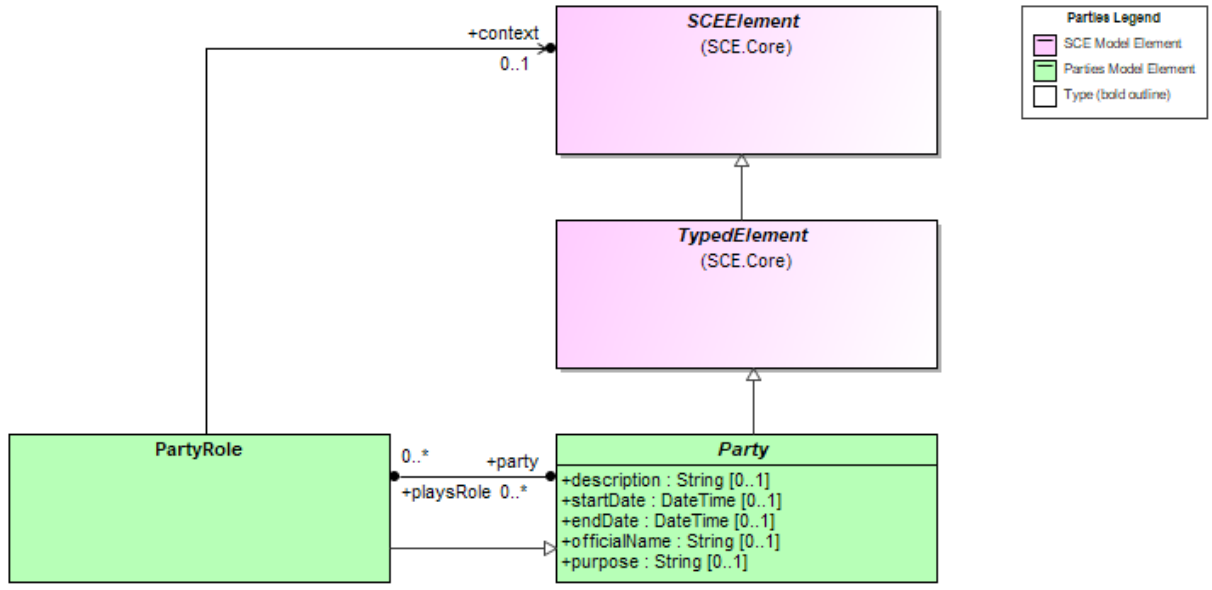


Figure 64: Party Role

This diagram shows the mapping of *Party* and its specializations to *PartyType* and its specializations.

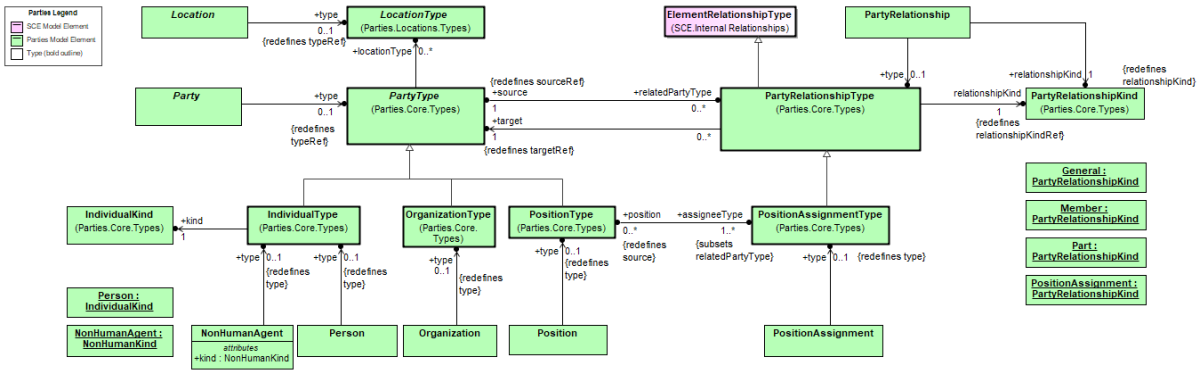


Figure 65: Parties and Party Types

10.1.1.1 Delegation

A kind of *PositionAssignment* relationship that states that one *Party* has been assigned a set of responsibilities by some authority.

Generalizations

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

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

Properties

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

Table 105. Delegation Attributes and/or Associations

Property/Association	Description
authority : Party [0..1]	The *Party* on whose authority the *Delegation* was made.
delegatedResponsibility : Position [1]	The responsibilities, stated as a *Role*, that are being delegated.
delegatee : Party [1]	The *Party* to whom the *Role* was delegated.

10.1.1.2 NonHumanAgent

Some type of automated system.

Generalizations

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

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

Properties

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

Table 106. NonHumanAgent Attributes and/or Associations

Property/Association	Description
kind : NonHumanKind []	An instance that indicates the kind of NonHumanAgent the element represents.
type : IndividualType [0..1]	The class that provides a specification of the *Automation*.

10.1.1.3 Organization

Organization is used to represent a group of *Parties.* The group may be a company, a department within a company, a club, a consortium, or some other group.

Generalizations

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

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

Properties

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

Table 107. Organization Attributes and/or Associations

Property/Association	Description
childRelationships : OrganizationStructureRelationship [0..*]	A set of relationships to the members of the *Organization*.
type : OrganizationType [0..1]	The class that provides a specification of the *Organization*.

10.1.1.4 OrganizationStructureRelationship

A specialization of PartyRelationship used to indicate internal structural relationships of a *Party*.

Generalizations

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

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

Properties

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

Table 108. OrganizationStructureRelationship Attributes and/or Associations

Property/Association	Description
child : Party [1]	The *Party* that is a member of the `organization`.
parent : Organization [1]	The *Organization* in which the *Party* is a member.

relationshipKind : PartyRelationshipKind [1] default: Part	The kind of structural relationship an Organization has with another Party.

10.1.1.5 Party

Party is an abstract concept representing a *Person*, *Role*, *Organization*, or other entity involved in some activity, interaction or endeavor.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

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

Table 109. Party Attributes and/or Associations

Property/Association	Description
assignment : PositionAssignment [0..*]	A relationship indicating a *Position* to which the *Party* has been assigned.
description : String [0..1]	A textual description of the *Party*.
endDate : DateTime [0..1]	The effective end date of the *Party*.
location : Location [0..*]	The location of the *Party*.
officialName : String [0..1]	The official name of the *Party*.
parentRelationship : OrganizationStructureRelationship [0..1]	A set of relationships to the *Organizations* in which the *Party* has membership.
playsRole : PartyRole [0..*]	The roles played by a *Party*.
primaryLocation : Location [0..1]	The primary location of the *Party*.
purpose : String [0..1]	The purpose of the *Party* with respect to the pedigree and/or provenance context.
relationships : PartyRelationship [0..*]	*PartyRelationships* in which the Party is involved.

startDate : DateTime [0..1]	The effective start date of the *Party*.
type : PartyType [0..1]	The class that provides a specification of the *Party*.

10.1.1.6 PartyRelationship

A kind of *ElementRelationshiip* that indicates a relationship between two *Parties*.

Generalizations

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

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

Properties

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

Table 110. PartyRelationship Attributes and/or Associations

Property/Association	Description
endDate : DateTime [0..1]	The effective end date of the relationship.
relationshipKind : PartyRelationshipKind [1]	The kind of relationship between two Parties.
sourceParty : Party [1]	The source *Party* of the relationship.
startDate : DateTime [0..1]	The effective start date of the relationship.
targetParty : Party [1]	The target *Party* of the relationship.
type : PartyRelationshipType [0..1]	The class that provide a specification of the *PartyRelationship*.

10.1.1.7 PartyRole

A role played by a *Party* in some context. For instance, a Buyer or a Supplier.

Generalizations

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

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

Properties

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

Table 111. PartyRole Attributes and/or Associations

Property/Association	Description
context : SCEElement [0..1]	The context in which the *Party* plays the role.
party : Party [0..*]	The *Party* that plays the role.
type : PartyRoleType [0..1]	The class that provides a specification of the *PartyRole*.

10.1.1.8 Person

An individual homo sapiens.

Generalizations

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

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

Properties

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

Table 112. Person Attributes and/or Associations

Property/Association	Description
fullName : String [0..*]	The full name of the *Person*.
type : IndividualType [0..1]	The class that provides a specification of the *Person*.

10.1.1.9 Position

A Position is a formally defined role in an *Organization* filled by some *Person*. *Positions* are often associated with a set of responsibilities in some context.

Examples of *Positions* include Chief Executive Officer or Technical Staff Member.

Generalizations

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

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

Properties

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

Table 113. Position Attributes and/or Associations

Property/Association	Description
positionAssignment : PositionAssignment [0..*]	A *PositionAssignment* that indicates the *Party* that fills the *Position*.
type : PositionType [0..1]	The class that provides a specification of the *Position*.

10.1.1.10 PositionAssignment

PositionAssignment indicates a *Party* is assigned to a particular *Position* for a particular period of time.

Generalizations

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

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

Properties

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

Table 114. PositionAssignment Attributes and/or Associations

Property/Association	Description
assignee : Party [1]	The *Party* that fills or filled the *Position*.
isUnique : Boolean [] default: false	A boolean stating whether only one *Party* filled a particular Role during that particular date range.
position : Position [1]	The *Position* filled by the noted *Party*.
relationshipKind : PartyRelationshipKind [1] default: PositionAssignment	The kind of relationship between an Organization and a Position within that Organization.
type : PositionAssignmentType [0..1]	The class that provides a specification of the *PositionAssignment*.

10.1.2 Types

The Core.Types section of the **Parties** metamodel contains elements related to the kinds of people, organizations, roles, automated systems and the relationships between them that are of interest in some context. These elements enable modeling kinds of Parties rather than particular Parties. Elements in the Core.Types section are generally specializations of **SCE** *ElementTypes* and as such provide a specification Parties to be created using elements in the Core.Instances section described above.

PartyTypes define the types or classifications of *Parties*. *PartyTypes* provide the ability to specify or "configure" organizational structures for different kinds of parties such as companies, non-profits, community organsations and many others. *PartyType* configurations can be used to provide a constraint mechanism on the *Parties* created in some context though the Party metamodel does not require their use.

While *PartyType* itself is abstract, the Party metamodel includes the concrete specializations *OrganizationType*, *IndividualType*, and *PositionType*. These types correspond to the concrete specializations of *Party* where

IndividualType is used as the type for *Person*, *Automation*, and *SoftwareAgent* with the `kind` property set appropriately,

PartyRelationshipTypes capture the possible relationships between *PartyTypes*. *PartyRelationshipTypes* have a *PartyRelationshipKind* that specifies the kind of relationship: Part, Member, Assignment, or General. (See *PartyRelationshipKind*.) *PositionAssignmentType* captures the particular relationship type wherein one or more *PartyTypes* are expected to fill (or be assigned to) a particular *PositionType*.

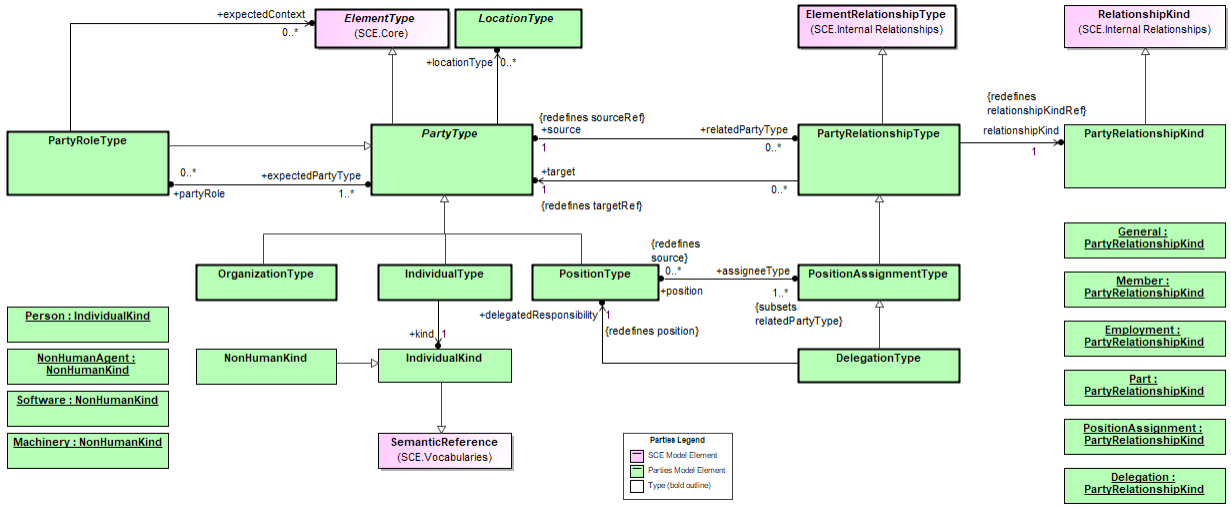


Figure 66: Party Types

PartyRoles define the types or classifications of the roles that may be played by one or more kinds of *Parties* (i.e., *PartyTypes*) in some context. The `expectedPartyType` property specifies which *PartyTypes* are expected to play *PartyRoles* of that *PartyRoleType*.

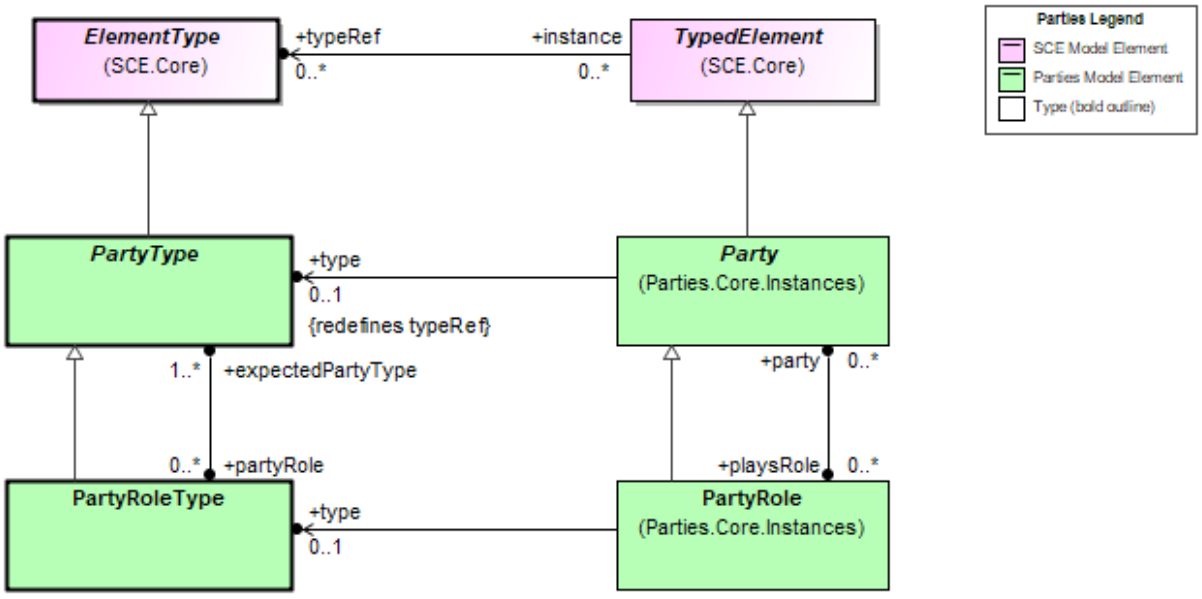


Figure 67: Party Role Type

Delegation captures the notion that a Party may assign a set of responsibilities to another party. *DelegationType* supports the ability to state that the responsibilities associated with a *PositionType* may be delegated to particular *PartyTypes* on the authority of some *PartyType*.

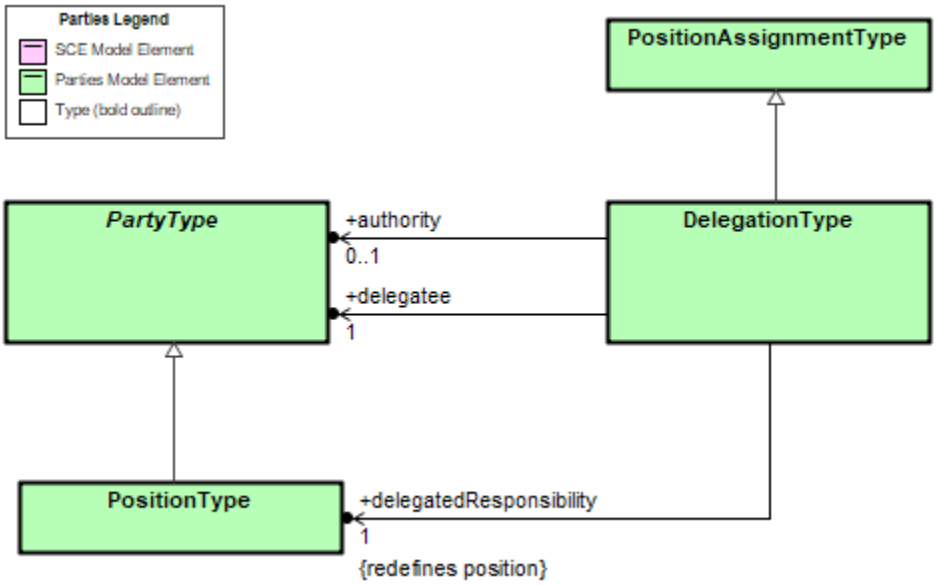


Figure 68: Delegation Types

10.1.2.1 DelegationType

DelegationType indicates a particular *PartyType* is delegated responsibility for particular *PositionType* by an authority.

Generalizations

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

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

Properties

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

Table 115. DelegationType Attributes and/or Associations

Property/Association	Description
authority : PartyType [0..1]	The *PartyType* expected to be the authority by which the delegation approved.
delegatedResponsibility : PositionType [1]	The set of responsibilities as defined by a *PositionType* that may be delegated.
delegatee : PartyType [1]	The *PartyType* to whom the responsibilities are expected to be delegated.

10.1.2.2 IndividualKind

IndividualKind is a kind of *SemanticReference* that serves as the foundation for `terms` in a *PartyVocabulary* that is used to specify the kinds of *IndividualTypes* in a **Parties** model. Instead of being defined a fixed enumerated list, the kinds are defined through instances of *IndividualKind* present in the *IndividualKinds* library. The instances defined in that library SHALL be included in any **Parties** implementation. However, the implementation can allow additional kinds of individuals through the addition of new instances of *IndividualKind* in the *IndividualKinds* library.

Generalizations

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

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

Properties

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

10.1.2.3 IndividualType

A kind of *PartyType* representing the type or classification of a *Party* of interest that is an individual such as a *Person*, *Automation*, or *SoftwareAgent*.

Generalizations

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

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

Properties

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

Table 116. IndividualType Attributes and/or Associations

Property/Association	Description
kind : IndividualKind [1]	An instance that indicates the kind of individual the *IndividualType* represents.

10.1.2.4 NonHumanKind

NonHumanKind is a kind of *IndividualKind* that serves as the foundation for `terms` in a *PartyVocabulary* that is used to specify the kinds of *NonHumanAgents* in a **Parties** model. Instead of being defined as a fixed enumerated list, the kinds are defined through instances of *NonHumanKind* present in the *IndividualKinds* library. The instances defined in that library SHALL be included in any **Parties** implementation. However, the implementation can allow additional kinds of individuals through the addition of new instances of *NonHumanKind* in the *IndividualKinds* library.

Generalizations

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

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

Properties

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

10.1.2.5 OrganizationType

A kind of *PartyType* that represents the type or classification of an *Organization*.

Generalizations

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

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

Properties

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

10.1.2.6 PartyRelationshipKind

PartyRelationshipKind is a specialization of *RelationshipKind* that serves as the foundation for `terms` for a *PartiesVocabulary* that is used to specify the kind of relationship that exists between two *PartyTypes* related by a *PartyRelationshipType*. Instead of being defined a fixed enumerated list, the kinds are defined through instances of *PartyRelationshipKind* present in the *PartyRelationshipKinds* library. The instances defined in the **Parties Library** SHALL be included in any **Parties** implementation. However, the implementation can allow additional kinds of relationship types through the addition of new instances of *PartyRelationshipKind* in the *PartyRelationshipKinds* library.

Generalizations

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

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

Properties

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

10.1.2.7 PartyRelationshipType

A kind of *ElementRelationshiip* that indicates a relationship between two *PartyTypes*.

Generalizations

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

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

Properties

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

Table 117. PartyRelationshipType Attributes and/or Associations

Property/Association	Description
relationshipKind : PartyRelationshipKind [1]	A specification of the kind of relationship of expected to exist between two Parties or PartyTypes.
source : PartyType [1]	The source *PartyType* of the relationship.
target : PartyType [1]	The target *PartyType* of the relationship.

10.1.2.8 PartyRoleType

A type or classification of a role that may be played by a particular *PartyType* in some context. For instance, a Buyer or a Supplier.

Generalizations

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

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

Properties

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

Table 118. PartyRoleType Attributes and/or Associations

Property/Association	Description
expectedContext : ElementType [0..*]	The context in which instances of the *PartyRoleType* are expected to occur.
expectedPartyType : PartyType [1..*]	The type of *Party* that is expected to play the role specified by the *PartyRoleType*.

10.1.2.9 PartyType

An abstract class representing the type or classification of a *Party* of interest.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

Table 119. PartyType Attributes and/or Associations

Property/Association	Description
locationType : LocationType [0..*]	The type of *Location* at which the instances of the *PartyType* are expected to be located.
partyRole : PartyRoleType [0..*]	The type(s) of roles that *Parties* of type *PartyType* are expected to play.
relatedPartyType : PartyRelationshipType [0..*]	The related *PartyType* of a relationship.

10.1.2.10 PositionAssignmentType

PositionAssignmentType indicates a particular *PartyType* is expected to fill particular *PositionType*.

Generalizations

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

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

Properties

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

Table 120. PositionAssignmentType Attributes and/or Associations

Property/Association	Description
kind : [1]	The kind relationship between the *PartyTypes* that is set to *Assignment*
position : PositionType [0..*]	The *PositionType* that will be filled by the *PartyType* referenced by the target of the *PositionTypeAssignment*.

10.1.2.11 PositionType

A kind of *PartyType* that represents the type or classification of a *Position*.

Generalizations

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

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

Properties

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

Table 121. PositionType Attributes and/or Associations

Property/Association	Description
assigneeType : PositionAssignmentType [1..*]	A *PositionAssignmentType* that indicates the *PartyType* that may fill the *PositionType*.

10.2 Locations

The Locations package contains elements related to physical or virtual locations.

10.2.1 Instances

The Locations.Intances section of the **Parties** metamodel contains elements related to locations and the relationships between them. These elements enable modeling specific locations at which Parties may reside. Elements in the Locations.Instances section are generally specializations of **SCE** *TypedElements* and as such may have an ElementType specified. The corresponding types are described below in the Locations.Types section.

Organizations may deem the location at which an occurrence took place to be of significance. In those situations a Location, either physical or virtual, may be captured in conjunction with an Occurrence.

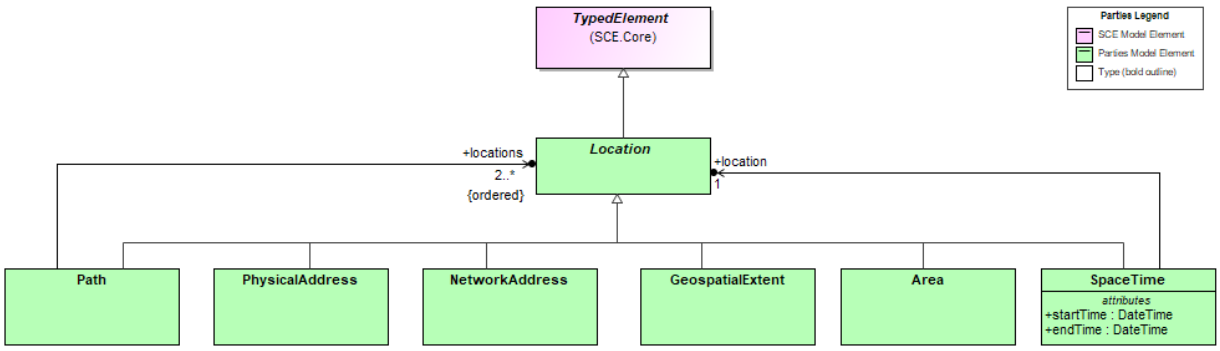


Figure 69: Locations

10.2.1.1 Area

A kind of location that encompasses some region in the world.

Generalizations

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

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

Properties

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

Table 122. Area Attributes and/or Associations

Property/Association	Description
type : AreaType [0..1]	The class that provides a specification of the *Area*.

10.2.1.2 GeospatialExtent

A location that is a volume in the world such as a container or a room.

Generalizations

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

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

Properties

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

Table 123. GeospatialExtent Attributes and/or Associations

Property/Association	Description
type : VolumeType [0..1]	The class that provides a specification of the *GeospatialExtent*.

10.2.1.3 Location

A particular place or position.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

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

Table 124. Location Attributes and/or Associations

Property/Association	Description
description : String [0..1]	A description of the *Location*.
type : LocationType [0..1]	The class that provides a specification of the *Location*.

10.2.1.4 NetworkAddress

The address of an element or node on a network.

Generalizations

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

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

Properties

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

Table 125. NetworkAddress Attributes and/or Associations

Property/Association	Description
type : NetworkAddressType [0..1]	The class that provides a specification of the *NetworkAddress*.

10.2.1.5 Path

An ordered collection of *Locations*.

Generalizations

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

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

Properties

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

Table 126. Path Attributes and/or Associations

Property/Association	Description
locations : Location [2..*]	The locations that specify the *Path*.
type : PathType [0..1]	The class that provides a specification of the *Path*.

10.2.1.6 PhysicalAddress

A physical location in the real world that has an identifiable address.

Generalizations

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

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

Properties

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

Table 127. PhysicalAddress Attributes and/or Associations

Property/Association	Description
type : PointType [0..1]	The class that provides a specification of the *PhyicalAddress*.

10.2.1.7 SpaceTime

A *Location* at a particular point in time.

Generalizations

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

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

Properties

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

Table 128. SpaceTime Attributes and/or Associations

Property/Association	Description
endTime : DateTime []	The ending time of the *SpaceTime*.
location : Location [1]	The location of the *SpaceTime*.
startTime : DateTime []	The starting time of the *SpaceTime*.
type : SpaceTimeType [0..1]	The class that provides a specification of the *SpaceTime*.

10.2.2 Types

The Locations.Types section of the **Parties** metamodel contains elements related to the kinds of locations and the relationships between them that are of interest in some context. These elements enable modeling kinds of Locations rather than particular Locations. Elements in the Locations.Types section are generally specializations of **SCE** *ElementTypes* and as such provide a specification of Locations to be created using elements in the Locations.Instances section described above.

10.2.2.1 AreaType

A kind of *LocationType* that states that a *Location* is a region or surface in the world.

Generalizations

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

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

Properties

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

10.2.2.2 LocationType

A class representing the type or classification of a *Location.*.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

10.2.2.3 NetworkAddressType

A class that specifies that *Locations* of this type are *NetworkAddresses.*

Generalizations

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

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

Properties

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

10.2.2.4 PathType

A kind of *LocationType* that states that a *Location* is a path.

Generalizations

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

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

Properties

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

10.2.2.5 PointType

A kind of *LocationType* that states that a *Location* is a specific point in the world.

Generalizations

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

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

Properties

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

10.2.2.6 SpaceTimeType

A kind of *LocationType* that states that a *Location* is a *Location* at a particular time.

Generalizations

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

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

Properties

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

10.2.2.7 VolumeType

A kind of *LocationType* that states that a *Location* is a volume in the world such as a container or room.

Generalizations

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

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

Properties

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

10.3 Packages

The Packages package provides elements to support the packaging of Parties-related elements.

The following figure presents the attributes and associations for the **Parties** packaging elements, including details about the elements they contain:

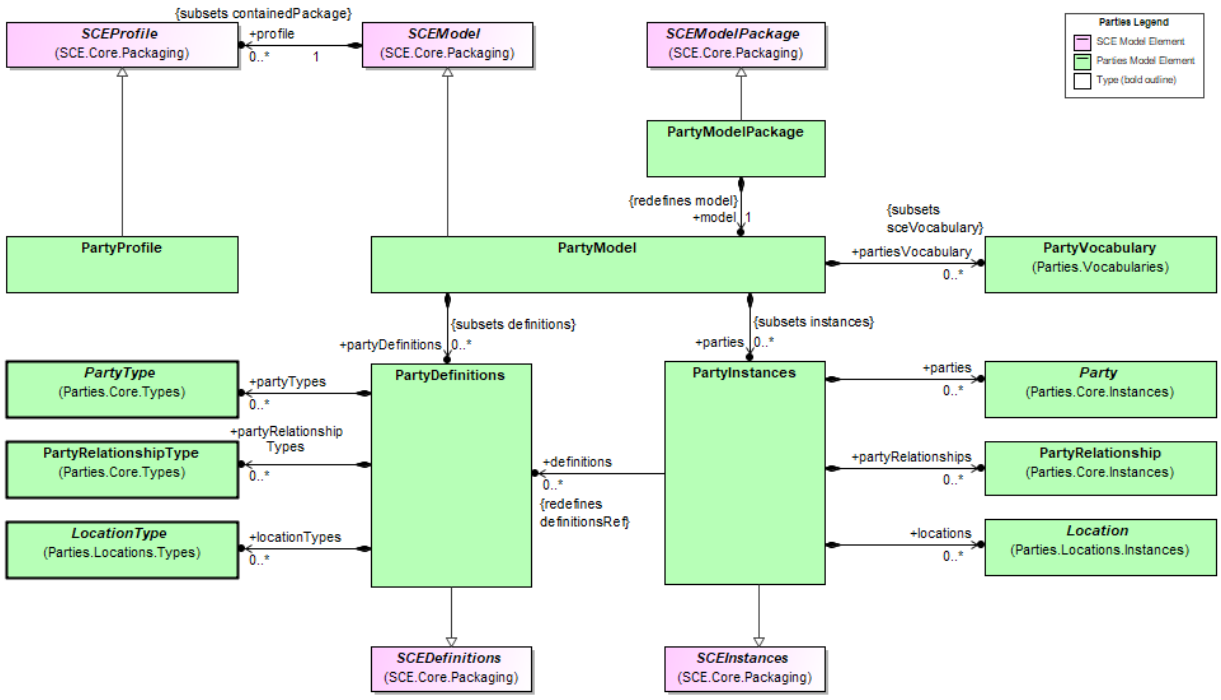


Figure 70: Party Packages

10.3.1 PartyDefinitions

PartyDefinitions is a kind of SCEDefinitions that contains the definitions of PartyTypes that are used to specify types of Party structures.

Generalizations

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

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

Properties

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

Table 129. PartyDefinitions Attributes and/or Associations

Property/Association	Description
locationTypes : LocationType [0..*]	The `locationTypes` property references the *LocationTypes* contained within the *PartyDefinitions* package.
partyRelationshipTypes : PartyRelationshipType [0..*]	The `partyRelationshipTypes` property references the *PartyRelationshipTypes* contained within the *PartyDefinitions* package.
partyTypes : PartyType [0..*]	The `partyTypes` property references the *PartyTypes* contained within the *PartyDefinitions* package.

10.3.2 PartyInstances

PartyInstances is kind of *SCEInstances* package that contains *Parties, PartyRelationships,* and their *Locations*.

Generalizations

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

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

Properties

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

Table 130. PartyInstances Attributes and/or Associations

Property/Association	Description
definitions : PartyDefinitions [0..*]	The property refers to zero or more *SCEDefinitions* packages that contains the *ElementTypes* that provide a basis for the instances contained in the *PartyInstances* package.
locations : Location [0..*]	The `locations` property references the *Location* elements contained within the *PartyInstances* package.
parties : Party [0..*]	The `parties` property references the *Party* elements contained within the *PartyInstances* package.
partyRelationships : PartyRelationship [0..*]	The `partRelationships` property references the *PartyRelationship* elements contained within the *PartyInstances* package.

10.3.3 PartyModel

PartyModel is kind of *SCEModel* that contains definitions of types of *Parties* as well as specifications of *Party* structures themselves.

Generalizations

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

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

Properties

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

Table 131. PartyModel Attributes and/or Associations

Property/Association	Description
parties : PartyInstances [0..*]	The `parties` property subsets the *SCEModel* `instances` property. It contains a list of all the *PartyInstance* sub-packages contained within a *SCEModel*.

partiesVocabulary : PartyVocabulary [0..*]	The `partiesVocabulary` is a list of terms (as *SemanticReferences*) that provide an extensible mechanism to define the elements of enumerations in a *PartiesModel*.
partyDefinitions : PartyDefinitions [0..*]	The `partyDefinitions` property subsets the *SCEModel* `definitions` property. It contains a list of all the *PartyDefinitions* sub-packages contained within a *PartyModel*.

10.3.4 PartyModelPackage

The *PartyModelPackage* is a specialization of *SCEModelPackage* and the main package for a Parties model. When the content of that model is serialized, the elements will be contained within a *PartyModelPackage*. PartyModelPackage SHALL contain one PartyModel as the `model` and zero or more PartiesDI packages as the `presentation`.

Further, as a specialization of *SCEPackage PartyModelPackages* may contain other *SCEPackages* and can import other *SCEPackages* as well,

Generalizations

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

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

Properties

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

Table 132. PartyModelPackage Attributes and/or Associations

Property/Association	Description
model : PartyModel [1]	The `model` property references the *PartyModel* contained within the PartyModelPackage. This is a subset of the `containedPackage` association of the *SCEPackage* element.

10.3.5 PartyProfile

A *PartyProfile* is a kind of *SCEProfile* that comprises profiles that can be applied to elements in a **PartyModel**. *PartyProfiles* provide a mechanism to exchange profile libraries.

Generalizations

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

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

Properties

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

10.4 Primitives

The *Primitives* package provides primitive data elements used by other **Parties** elements.

The following figure presents the primitive elements used in the **Parties** metamodel:

Figure 71: Primitives

10.4.1 DateTime

A primitive that captures a point in time including a date and the time of day to greatest precision practical.

Generalizations

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

Properties

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

10.5 Vocabularies

PartyVocabularies are sets of terms used within a **Parties** model that are defined by an external ontology. The terms link to formal definitions for the terms used within the model. The *SemanticReference* element is used to name the term provide a link to the definitions. *PartyVocabularies* are contained within an *PartiesModel* package.

The following figure presents the elements related to the *PartyVocabulary* section:

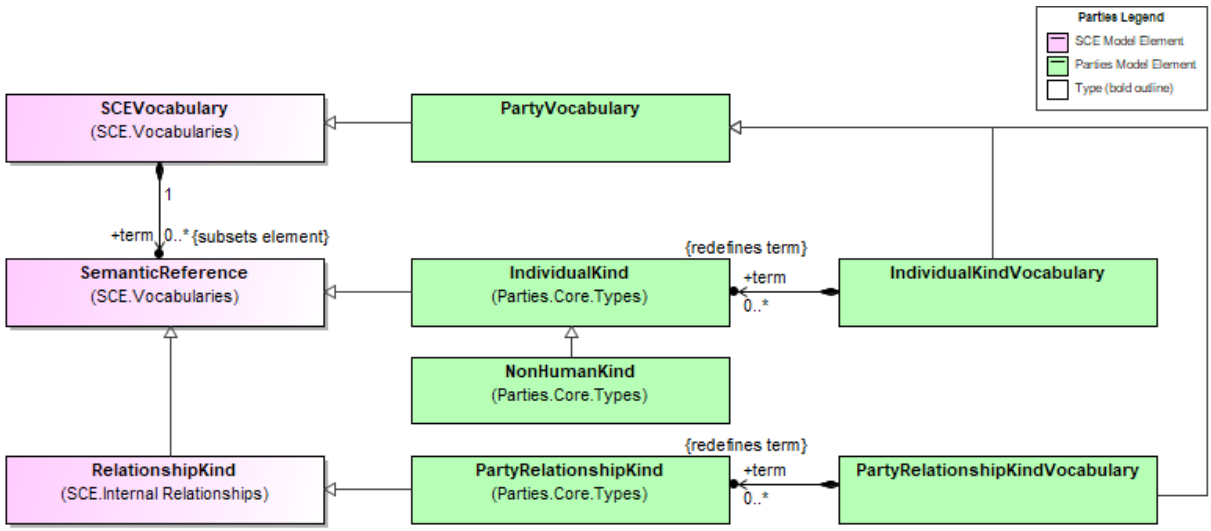


Figure 72: PartyVocabularies

10.5.1 PartyVocabulary

A *PartyVocabulary* is a kind of *SCEVocabulary* that includes a list of terms defined as instances of the *SemanticReference* element. As instances of SemanticReference, or a specialization thereof, the instances can be used to relate the terms to external definitions of the meaning of the term. The terms themselves do not represent the

definitions or meanings but provide links to an external source. The **Parties** model contains two vocabularies: *PartyRelationshipKinds* and *IndividualKinds.*

Generalizations

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

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

Properties

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

10.5.2 IndividualKindVocabulary

A *IndividualKindVocabulary* is a kind of *PartiesVocabulary* that includes a list of terms defined as instances of *IndiviualKind*, itself a *SemanticReference*. As instances of a specialization of *SemanticReference*, the instances can be used to relate the terms to external definitions of the meaning of the term. The terms themselves do not represent the definitions or meanings but provide links to an external source.

Generalizations

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

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

Properties

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

Table 133. IndividualKindVocabulary Attributes and/or Associations

Property/Association	Description
term : IndividualKind [0..*]	A list of the terms representing valid IndividualKinds.

10.5.3 PartyRelationshipKindVocabulary

A *PartyRelationshipKindVocabulary* is a kind of *PartiesVocabulary* that includes a list of terms defined as instances of *PartyRelationshipKind*, itself a kind of *SemanticReference*. As instances of a specialization of *SemanticReference*, the instances can be used to relate the terms to external definitions of the meaning of the term. The terms themselves do not represent the definitions or meanings but provide links to an external source.

Generalizations

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

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

Properties

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

Table 134. PartyRelationshipKindVocabulary Attributes and/or Associations

Property/Association	Description
term : PartyRelationshipKind [0..*]	A list of the terms representing valid PartyRelationshipKinds.

11 Parties Library

A Library is included in the **Parties** specification to provide standard values that that are intended to be provided by tools implementing the **Parties** specification. Currently, **Parties** defines the standard values for two vocabularies: *IndividualKinds* and *PartyRelationshipKinds* (See next sections).

11.1 IndividualKinds

The *IndividualKinds* package contains the instances representing the standard *IndividualKinds* vocabulary. This vocabulary provides a standard set of terms for the kinds of Individuals that can be instantiated within a Parties model. These elements include an instance of a *PartiesVocabulary,* `IndividualKinds`, which represents the vocabulary itself as well as instances of *IndividualKind* representing the kinds of Individuals that may be instantiated.

The *IndividualKind* element is used to indicate a specific kind IndividualType that is to be created. The instances defined in this Library SHALL be included in any **Parties** implementation. However, the implementation can allow additional instances of the class to represent new IndividualTypes.

The following figure presents the instances for the *IndividualKind* element that are `terms` for the `IndividualKinds` vocabulary.

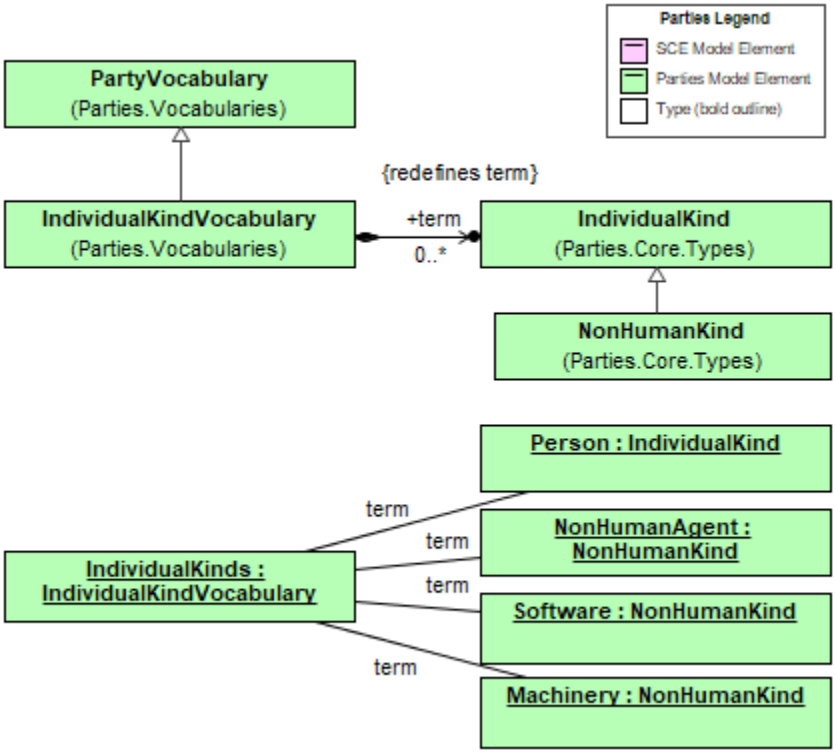


Figure 73: IndividualKinds

The following table provides a definition of the terms included in the *IndividualKinds* Vocabulary.

Table 135. IndividualKinds Vocabulary

#	Name	Documentation
1	IndividualKinds	`IndividualKinds` is an instance of *PartiesVocabulary* that includes terms for the kinds of *PartyRelationships* that may be created in a **Parties** model.
2	Machinery	`Machinery` indicates that the type of NonHumanKind is a machine of some kind.
3	NonHumanAgent	`NonHumanAgent` indicates that the type of individual is an automated system of some kind.
4	Person	`Person` indicates that the type of individual is a person.
5	Software	`Software` indicates that the type of individual is a software module of some kind.

11.1.1 IndividualKinds

`IndividualKinds` is an instance of *PartiesVocabulary* that includes terms for the kinds of *PartyRelationships* that may be created in a **Parties** model.

11.1.2 Machinery

`Machinery` indicates that the type of NonHumanKind is a machine of some kind.

11.1.3 NonHumanAgent

`NonHumanAgent` indicates that the type of individual is an automated system of some kind.

11.1.4 Person

`Person` indicates that the type of individual is a person.

11.1.5 Software

`Software` indicates that the type of individual is a software module of some kind.

11.2 PartyRelationshipKinds

The *PartyRelationshipKinds* package contains one instance of an *SCEVocabulary*: `PartyRelationshipKind` which is provided by the **Parties** Library. The purpose of this vocabulary is to provide a set of standard terms for the different types of relationships between Parties. These terms will be represented by instances of the *PartyRelationshipKind* element.

The instances defined in this Library SHALL be included in any **Parties** implementation. However, the implementation can allow additional instances of the class if required for a particular modeling situation. Specifying the kinds of Party relationships using this instantiation mechanism rather than a fixed enumerated list enables extension of the kinds of relationships that are possible without having to modify the standard.

The following figure presents the instances for the *PartyRelationshipKind* element that are `terms` for the instance (`PartyRelationshipKinds`) of the *PartiesVocabulary* element.

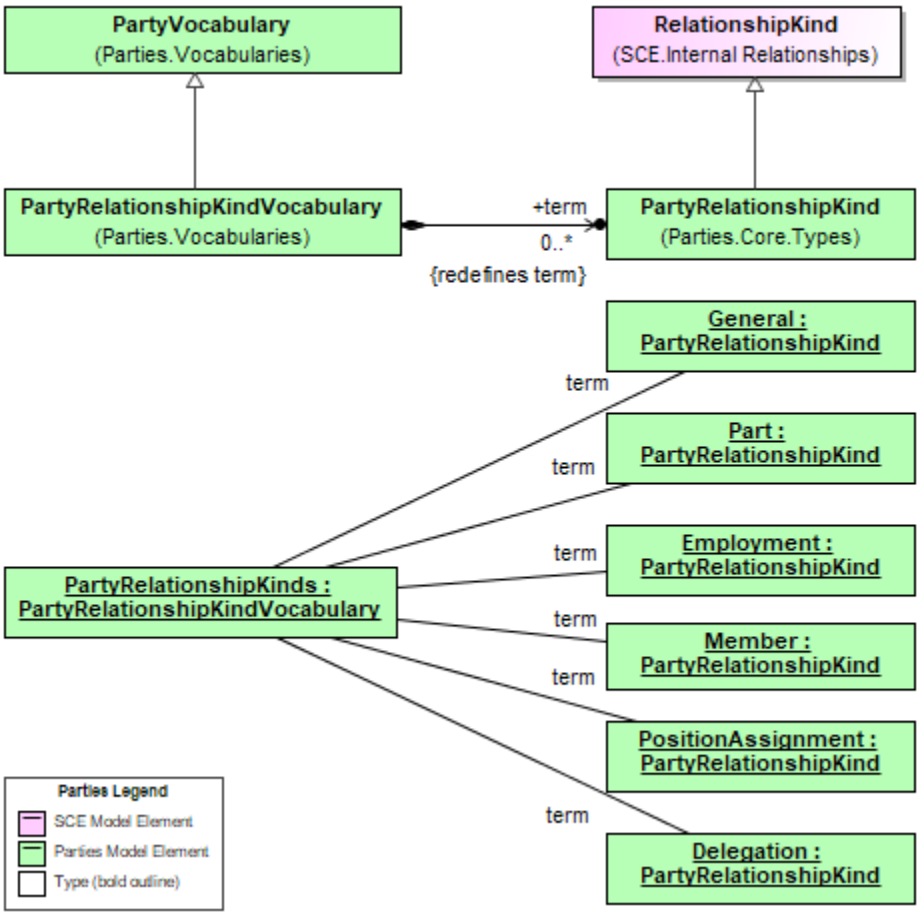


Figure 74: PartyRelationshipKinds

The following table provides a definition of the terms included in the *PartyRelationshipKinds* Vocabulary.

Table 136. PartyRelationshipKinds Vocabulary

#	Name	Documentation
1	PartyRelationshipKinds	`PartyRelationshipKinds` is an instance of *PartiesVocabulary* that includes terms for the kinds of *PartyRelationships* that may be created in a **Parties** model.
2	Delegation	`Delegation` indicates that the `target` element of the *PartyRelationship*, either a *Party* or *PartyType* has been delegated the responsibilities associated with the `source` element, either a *Position* or *PositionType*, respectively.
3	Employment	`Employment` indicates that the `targetParty` element of the *PartyRelationship* is employed by the `sourceParty`.

#	Name	Documentation
4	General	`General` indicates the existence of some general relationship between the `source` element of the *PartyRelationship* is a member of the `target` element.
5	Member	`Member` indicates that the `target` element of the *PartyRelationship* is a member of the `source` element.
6	Part	`Part` indicates that the `target` element of the *PartyRelationship* is a part of the `source` element.
7	PositionAssignment	`Assignment` indicates that the `source` element of the *PartyRelationship*, either a *Party* or *PartyType* is assigned to the `target` element, either a *Position* or *PositionType*, respectively.

11.2.1 PartyRelationshipKinds

`PartyRelationshipKinds` is an instance of *PartiesVocabulary* that includes terms for the kinds of *PartyRelationships* that may be created in a **Parties** model.

11.2.2 Delegation

`Delegation` indicates that the `target` element of the *PartyRelationship*, either a *Party* or *PartyType* has been delegated the responsibilities associated with the `source` element, either a *Position* or *PositionType*, respectively.

11.2.3 Employment

`Employment` indicates that the `targetParty` element of the *PartyRelationship* is employed by the `sourceParty`.

11.2.4 General

`General` indicates the existence of some general relationship between the `source` element of the *PartyRelationship* is a member of the `target` element.

11.2.5 Member

`Member` indicates that the `target` element of the *PartyRelationship* is a member of the `source` element.

11.2.6 Part

`Part` indicates that the `target` element of the *PartyRelationship* is a part of the `source` element.

11.2.7 PositionAssignment

`Assignment` indicates that the `source` element of the *PartyRelationship*, either a *Party* or *PartyType* is assigned to the `target` element, either a *Position* or *PositionType*, respectively.

12 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.

12.1 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:

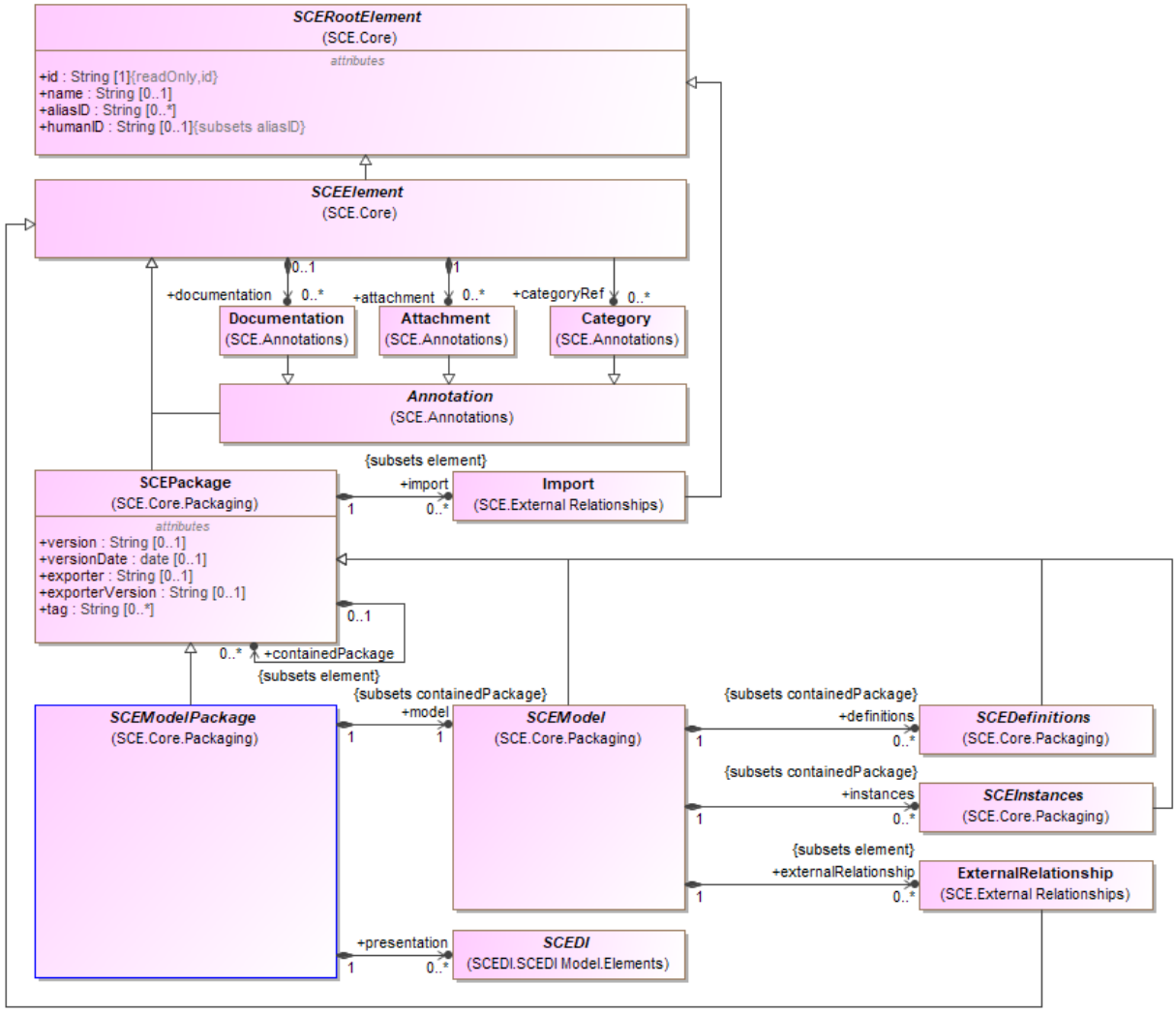


Figure 75: The SCE Core Structure Metamodel

12.1.1 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 137. SCERootElement Attributes and/or Associations

Property/Association	Description
aliasID : String [0..*]	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.
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.

12.1.2 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*:

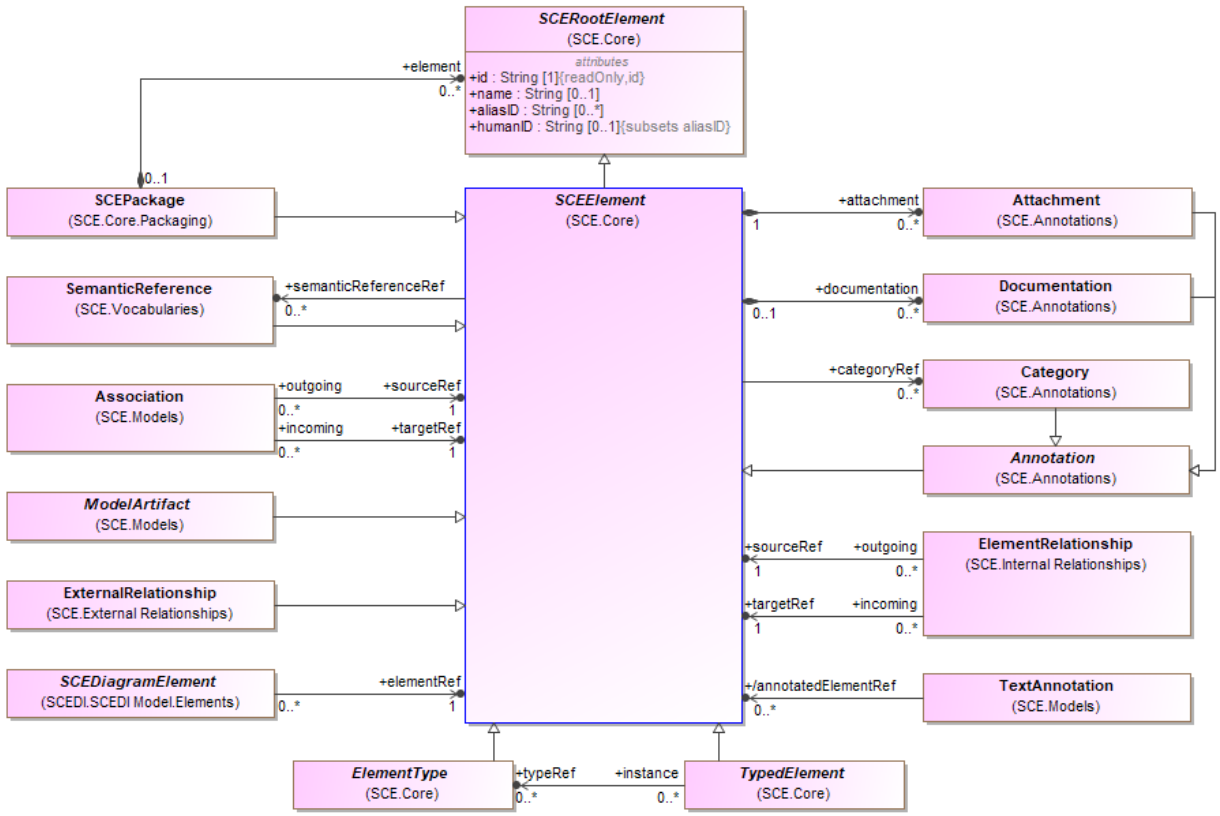


Figure 76: The SCEElement Metamodel

Generalizations

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

- **SCE** *SCERootElement* (see the section **SCE** specification for more information).

Properties

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

Table 138. SCEElement Attributes and/or Associations

Property/Association	Description
attachment : Attachment [0..*]	This association is used to annotate any concrete specialization of *SCEElement* with descriptions and other documentation.
categoryRef : Category [0..*]	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.
documentation : Documentation [0..*]	This association is used to annotate any concrete specialization of *SCEElement* with descriptions and other documentation.

extensionDefinitionRef : ExtensionDefinition [0..*]	This association is used to attach additional attributes and associations to any concrete specialization of *`SCEElement`* – i.e., provide an extension. This association is not applicable when the XML schema interchange is used, since the XSD mechanisms for supporting `anyAttribute` and any element already satisfy this requirement.
extensionValue : ExtensionAttributeValue [0..*]	This association is used to provide values for extended attributes and model associations. This association is not applicable when the XML schema interchange is used, since the XSD mechanisms for supporting `anyAttribute` and any element already satisfy this requirement.
semanticReferenceRef : SemanticReference [0..*]	A concrete *SCEElement* can reference zero or more *SemanticReference* elements.

12.1.3 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:

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

12.1.4 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:

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 139. TypedElement Attributes and/or Associations

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

12.1.5 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:

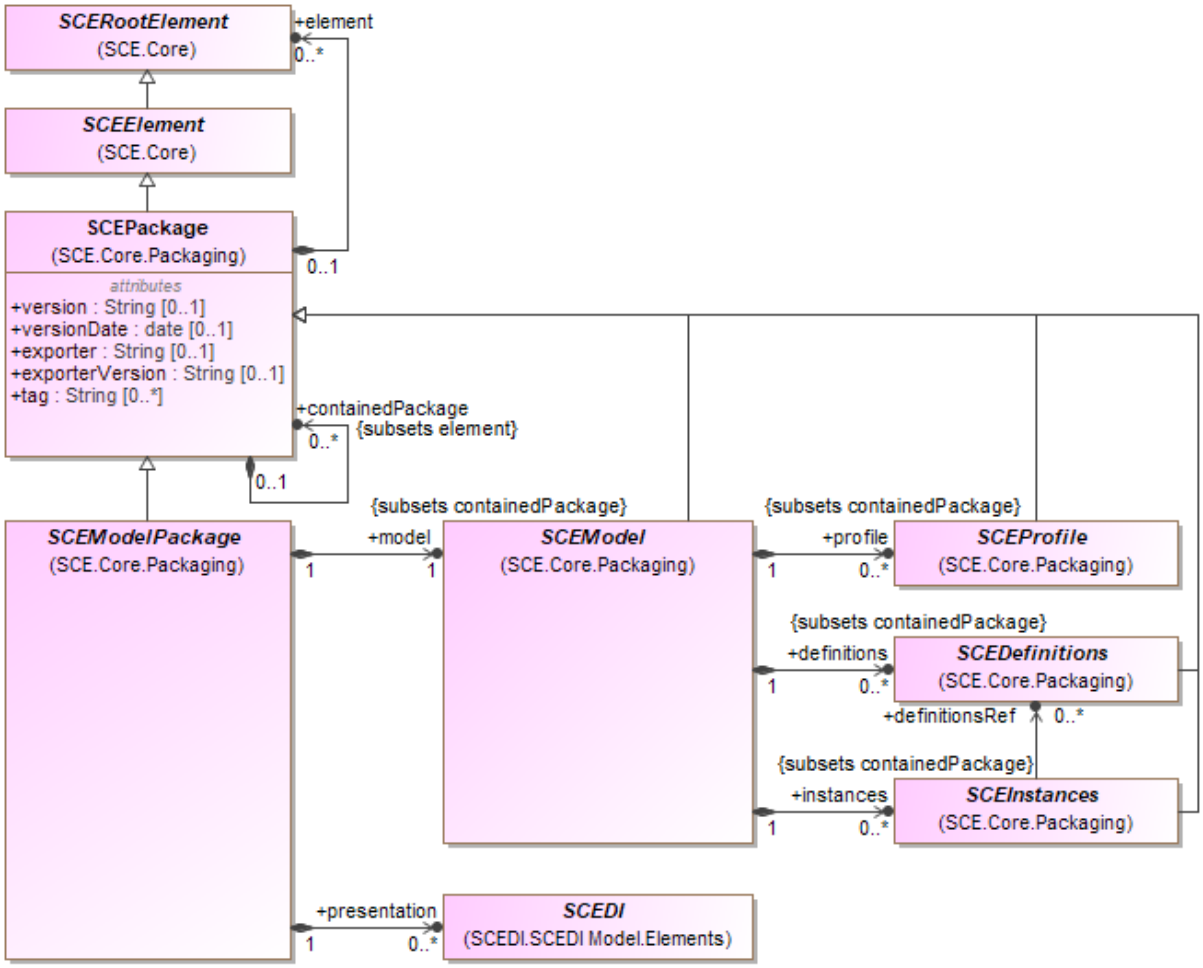


Figure 77: The SCE Packaging Elements Metamodel

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

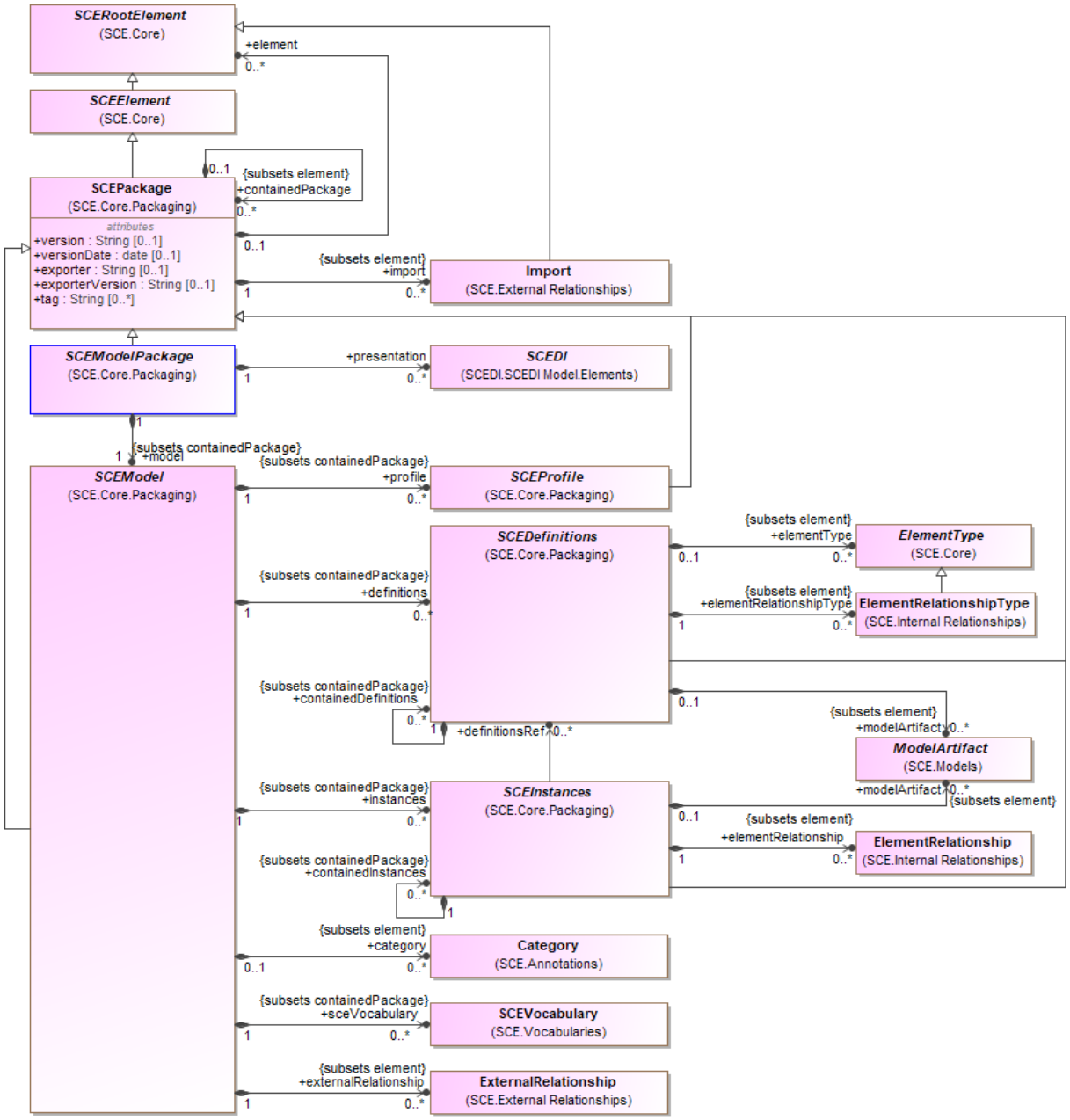


Figure 78: The SCE Packaging Elements Metamodel (Details)

12.1.5.1 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 `targetNamespce` attribute is not required for the metamodel elements for **SCE**. However, for non-XMI XSDs, a `targetNamespace` attribute of type `anyURI` will be included in the tSCEPackage type for the **SCE** XSD.

The following figure presents the metamodel for *SCEPackage*:

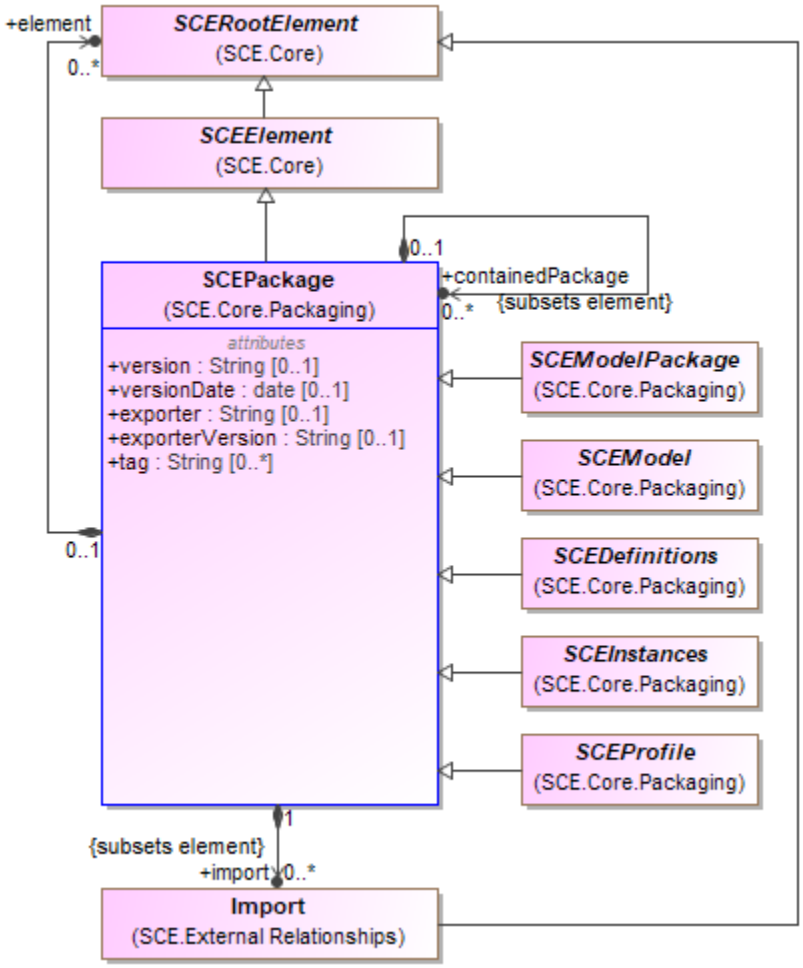


Figure 79: The SCEPackage Metamodel

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 140. SCEPackage Attributes and/or Associations

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

12.1.5.2 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 concreate modeling package.

The following figure presents the metamodel for *SCEModelPackage*:

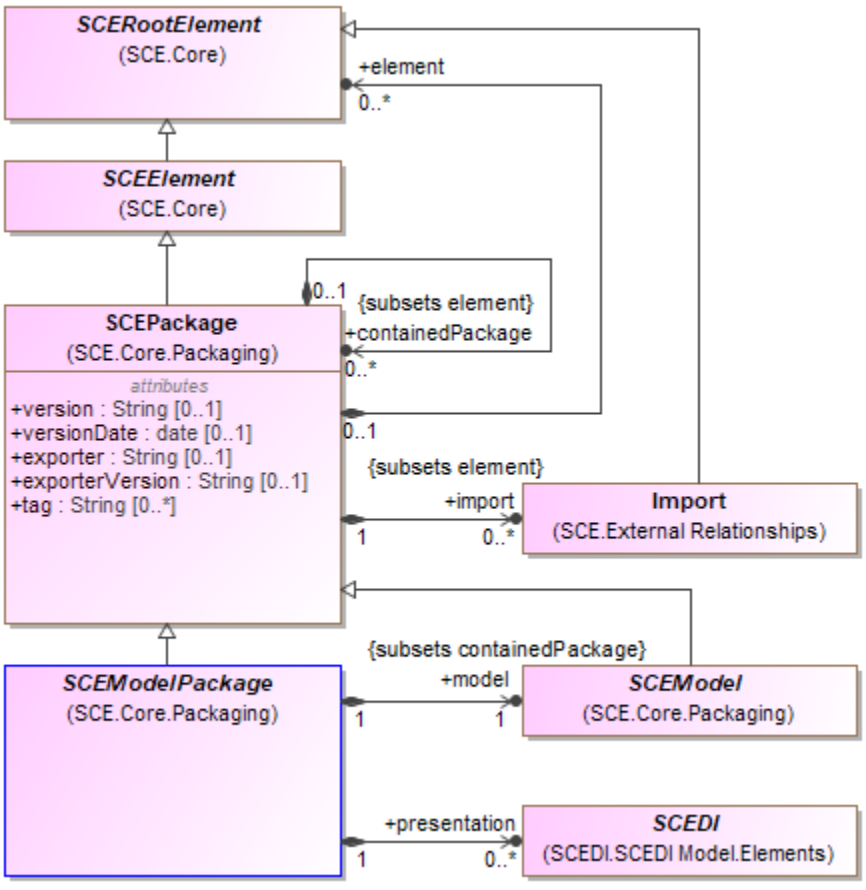


Figure 80: The SCEModelPackage Metamodel

Generalizations

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

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

Properties

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

Table 141. SCEModelPackage Attributes and/or Associations

Property/Association	Description
model : SCEModel [1]	This the *SCEModel* sub-package contained within a *SCEModelPackage*. This is a subset of the `containedPackage` association of the *SCEPackage* element.
presentation : SCEDI [0..*]	This attribute contains the Diagram Interchange information contained within this *SCEModelPackage*.

12.1.5.3 SCEModel

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

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

The following figure presents the metamodel for *SCEModel*:

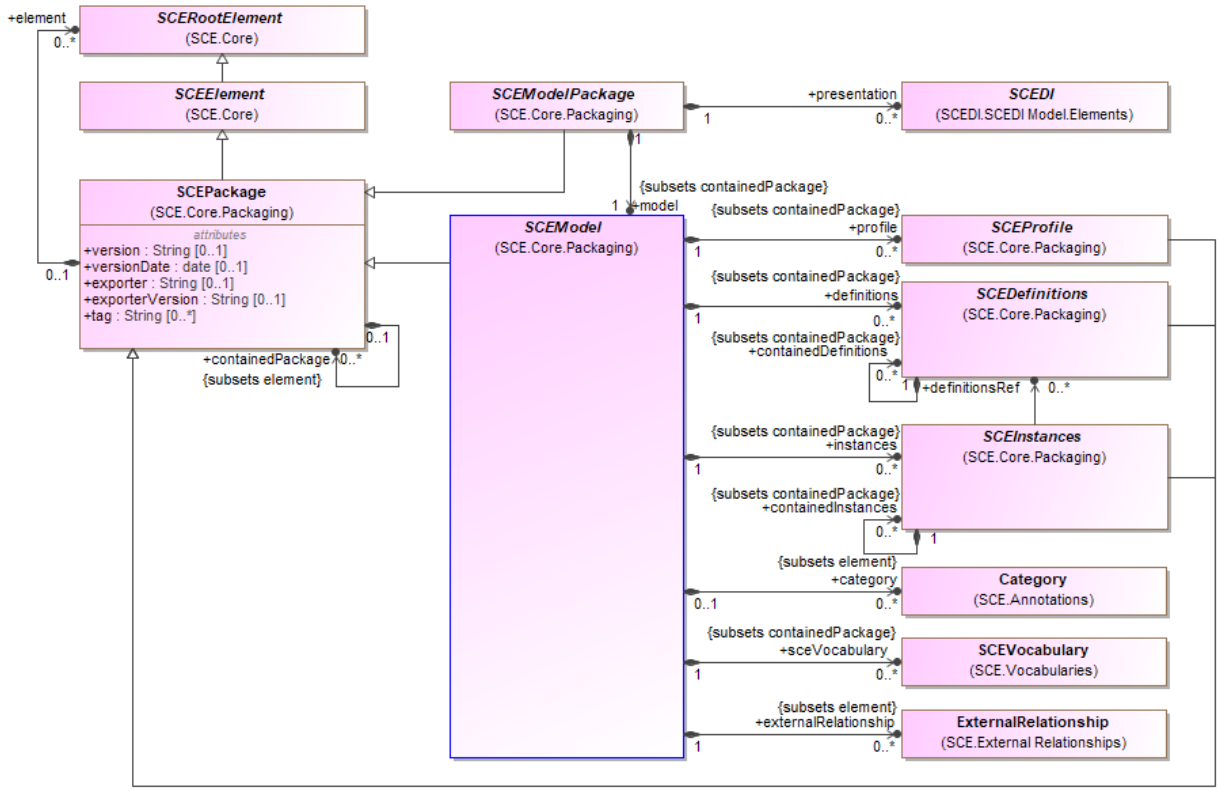


Figure 81: The SCEModel Metamodel

Generalizations

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

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

Properties

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

Table 142. SCEModel Attributes and/or Associations

Property/Association	Description
category : Category [0..*]	This is a list of all the *Categories* contained within a concrete specialization of *SCEModel*.
definitions : SCEDefinitions [0..*]	This is a list of all the *SCEDefinitions* sub-packages contained within a *SCEModel*. This is a subset of the `containedPackage` association of the *SCEPackage* element.
externalRelationship : ExternalRelationship [0..*]	This is a list of all the *ExternalRelationships* contained within a concrete specialization of *SCEModel*.

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

12.1.5.4 SCEDefinitions

The *SCEDefinitions* element is the package that, when specialized by a downstream language, will contain the "modeling" elements of that language.

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

The following figure presents the metamodel for *SCEDefinitions*:

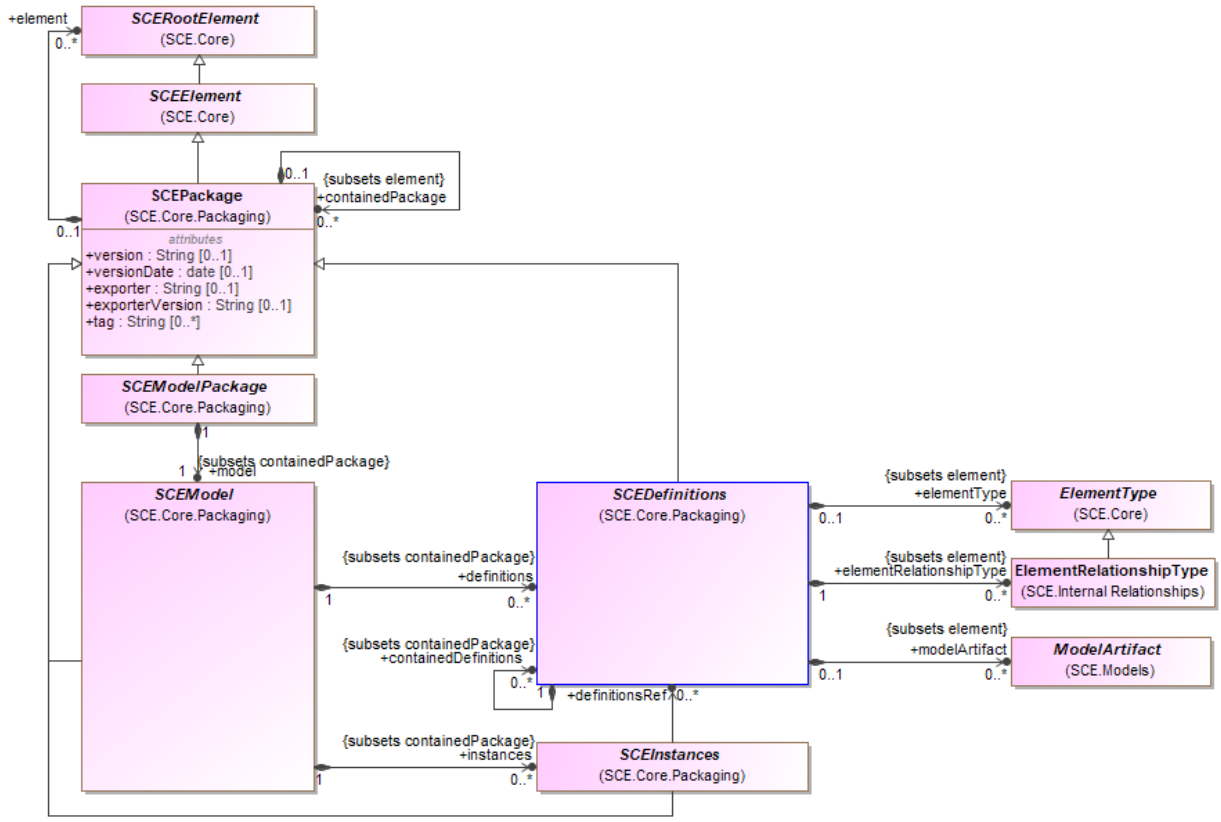


Figure 82: The SCEDefinitions Metamodel

Generalizations

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

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

Properties

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

Table 143. SCEDefinitions Attributes and/or Associations

Property/Association	Description
containedDefinitions : SCEDefinitions [0..*]	This is a list of all the sub-package *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.
elementRelationshipType : ElementRelationshipType [0..*]	This is a list of all the *ElementTypeRelationships* contained within a concrete specialization of *SCEDefinitions*. This is a subset of the `element` association of the *SCEPackage* element.
elementType : ElementType [0..*]	This is a list of all the *ElementTypes* contained within a *SCEDefinitions*. This is a subset of the `element` association of the *SCEPackage* element.
modelArtifact : ModelArtifact [0..*]	This is a list of all the *ModelArtifacts* contained within a concrete specialization of *SCEDefinitions*. These will usually be contained in an *SCEDefinitions* that is sub-package to the top-level *SCEDefinitions.* This is a subset of the `element` association of the *SCEPackage* element.

12.1.5.5 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 (e.g., the Provenance and Pedigree Model and Notation (**PPMN** – OMG Document bmi/21-02-03).

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

The following figure presents the metamodel for *SCEInstances*:

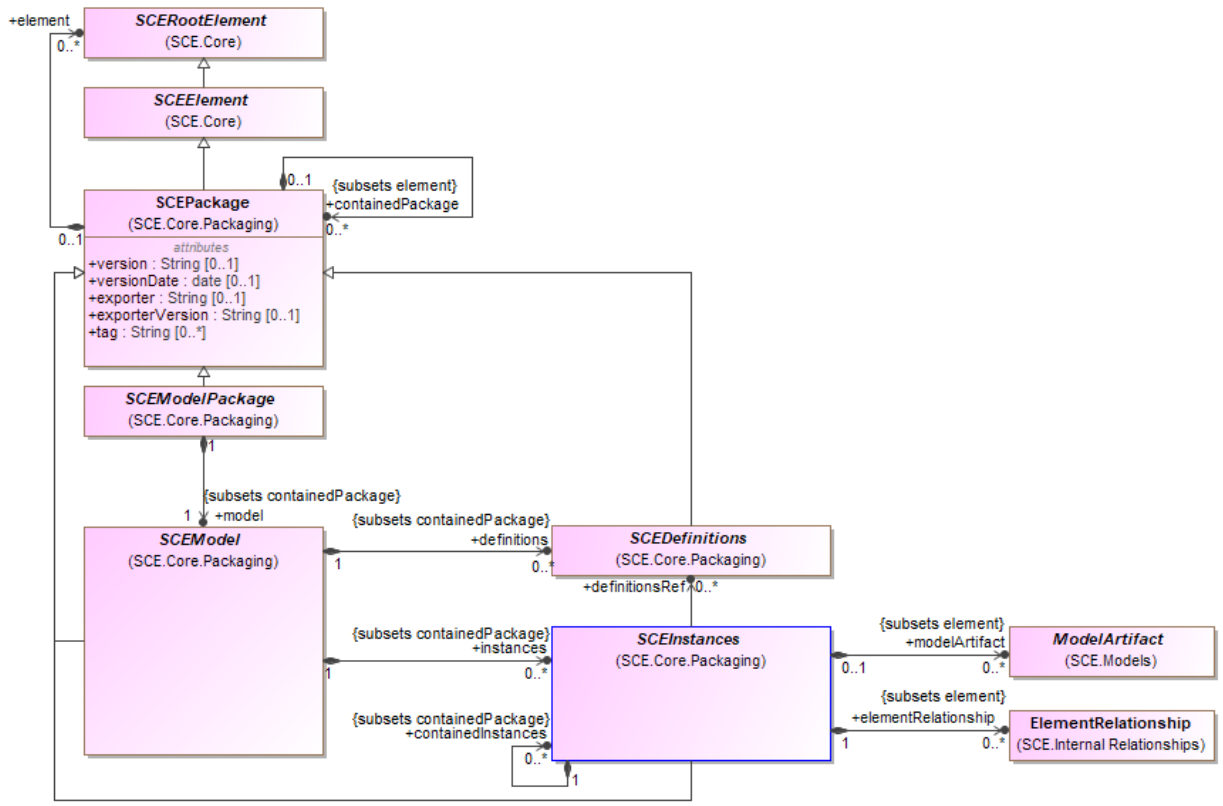


Figure 83: The SCEInstances Metamodel

Generalizations

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

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

Properties

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

Table 144. SCEInstances Attributes and/or Associations

Property/Association	Description
containedInstances : SCEInstances [0..*]	This is a list of all the sub-package *SCEInstances*. This provides the capability for all specializations of *SCEInstances* to include sub-packages. This is a subset of the `containedPackage` association of the *SCEPackage* element.
definitionsRef : SCEDefinitions [0..*]	This is a reference to an *SCEDefinitions* package that contains the *ElementType* elements that provide a basis for the instances contained in the *SCEInstances* package. Note that an *SCEInstances* package is not required to reference a *SCEDefinitions* package.

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

12.1.5.6 SCEProfile

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

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

Generalizations

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

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

Properties

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

12.2 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 to 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*.

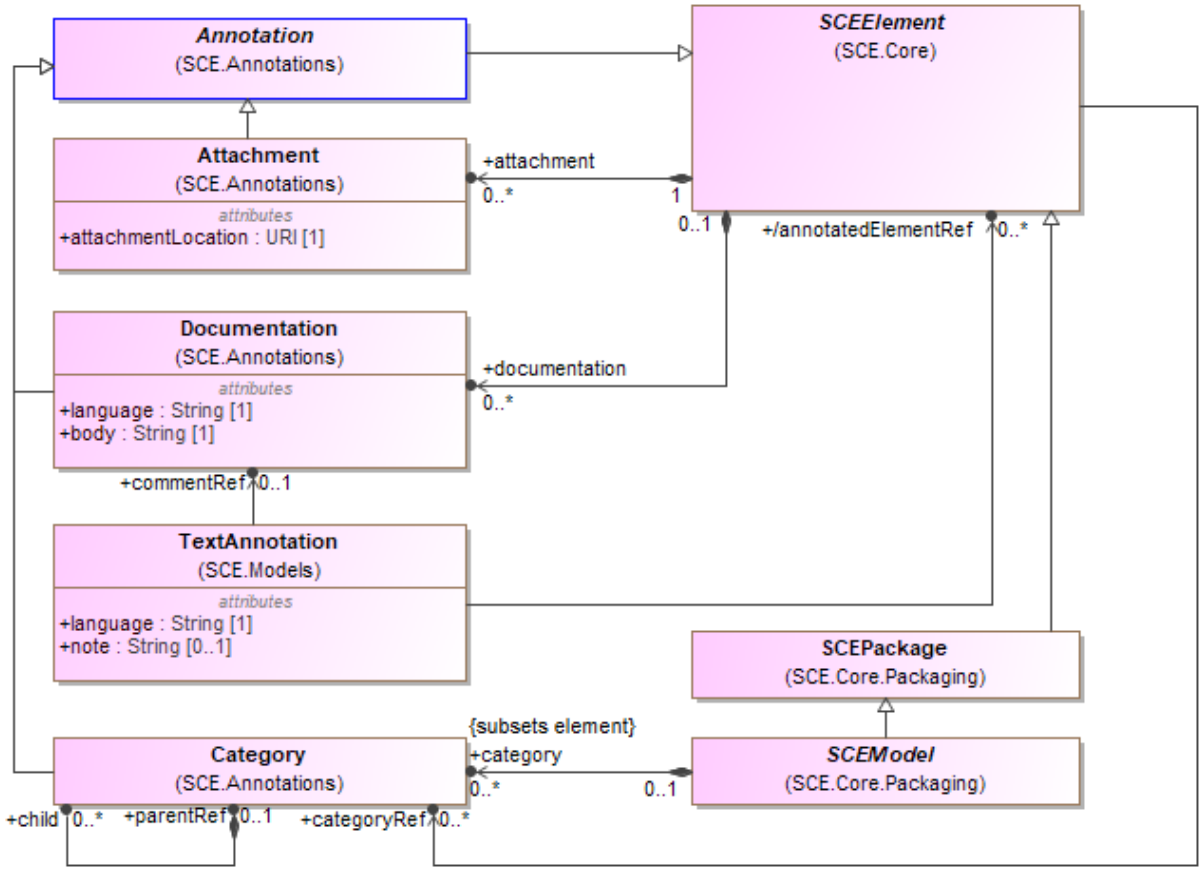


Figure 84: Annotations

12.2.1 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:

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

12.2.2 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).

Properties

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

Table 145. Attachment Attributes and/or Associations

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

12.2.3 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 *SCEPackage* (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*.

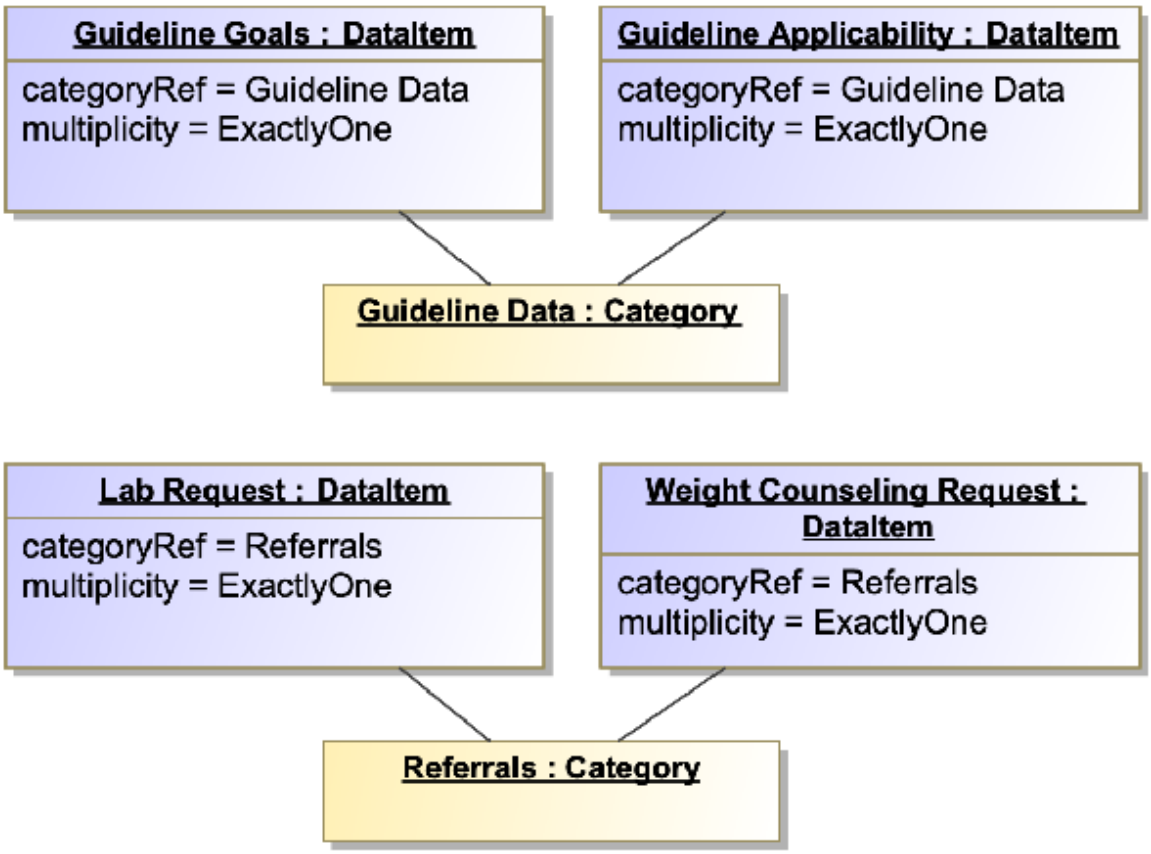


Figure 85: An Example of a Groups referencing Categories (in an UML Object Diagram)

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

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

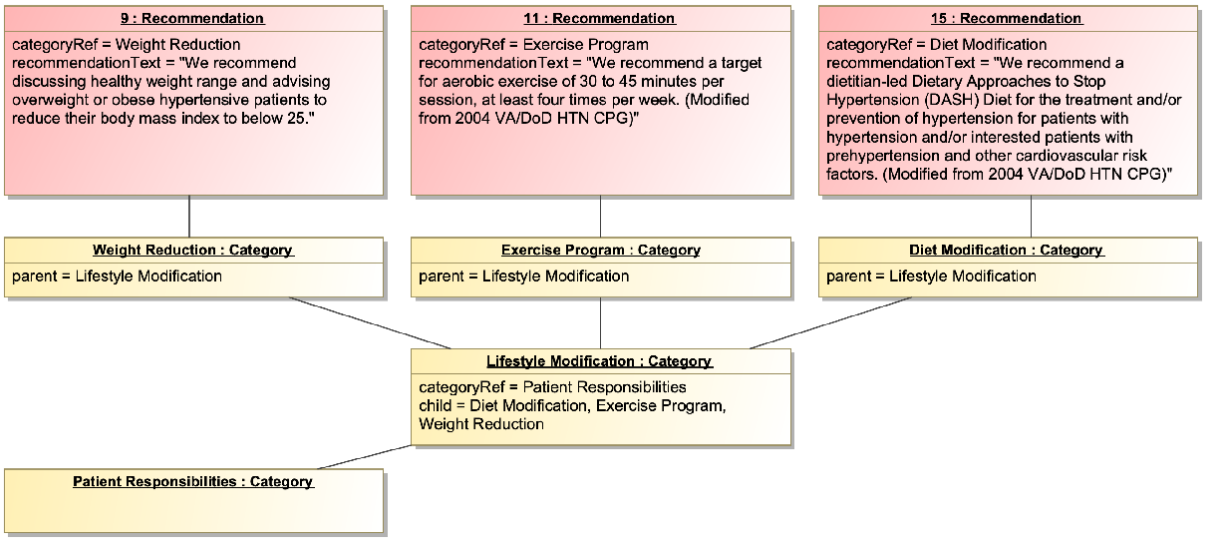


Figure 86: An Example of a Parent and Children Categories (in an UML Object Diagram)

Generalizations

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

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

Properties

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

Table 146. Category Attributes and/or Associations

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

12.2.4 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).

Properties

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

Table 147. Documentation Attributes and/or Associations

Property/Association	Description
body : String [1]	This attribute is used to capture the text descriptions of any concrete element within a model that is dependent on **SCE**.
language : String [1]	The named `language` can be a natural language, in which case the `body` is an informal representation, or an artifical language, in which case the `body` is expected to be a formal, machine-parsable representation.

12.3 External Relationships

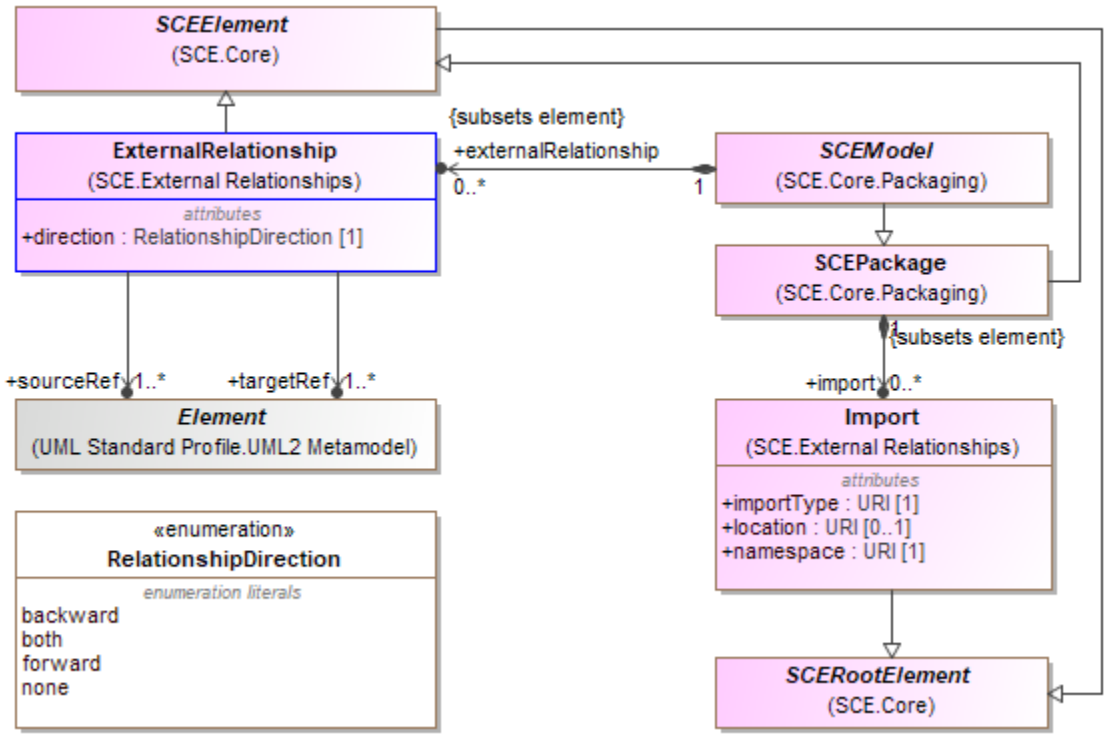


Figure 87: The External Relationships Metamodel

12.3.1 ExternalRelationship

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

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 148. ExternalRelationship Attributes and/or Associations

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

12.3.2 RelationshipDirection

This enumeration list specifies the direction of the relationship.

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

Table 149. RelationshipDirection Literals

Literal	Description
backward	This literal specifies that the *ExternalRelationship* is in the direction from the `target` to the `source`.
both	This literal specifies that the *ExternalRelationship* is in the direction from the `target` to the `source` and from the `source` to the `target`.
forward	This literal specifies that the *ExternalRelationship* is in the direction from the `source` to the `target`.
none	This literal specifies that the *ExternalRelationship* is in the direction from the `target` to the `source`.

12.3.3 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:

- **SCE** *SCERootElement* (see the section **SCE** specification for more information).

Properties

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

Table 150. Import Attributes and/or Associations

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

12.4 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*):

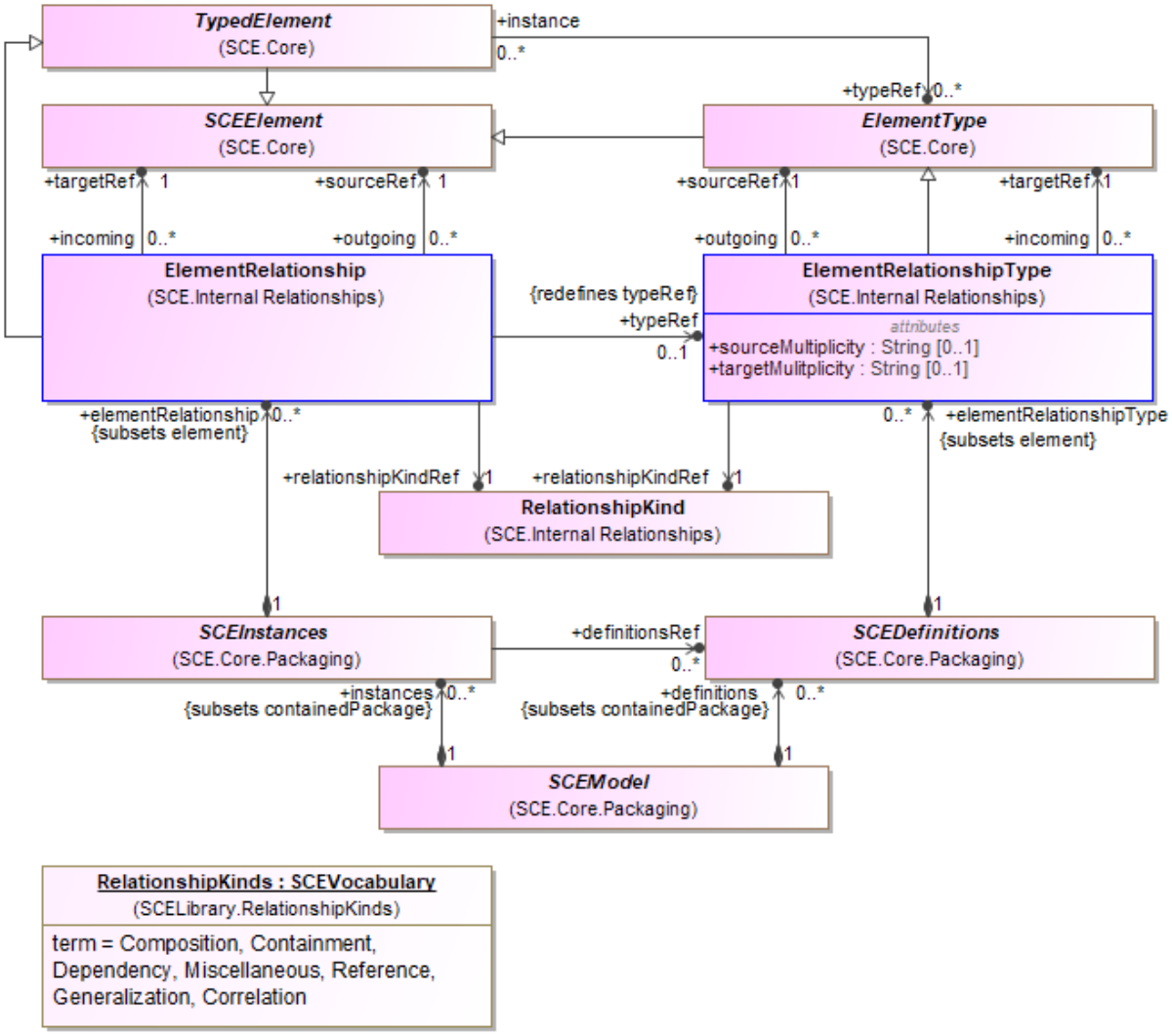


Figure 88: The Element Relationship Metamodel

12.4.1 ElementRelationship

A kind of relationships between two *SCEElements*. The *RelationshipType* enumeration element identify specific types of relationships.

Generalizations

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

- **SCE** *TypedElement* (see the section **SCE** specification for more information).

Properties

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

Table 151. ElementRelationship Attributes and/or Associations

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

12.4.2 ElementRelationshipType

A kind of *ElementRelationship* that specifies two *ElementTypes* (rather than *SCEElements*). The *RelationshipType* enumeration element identify specific types of relationships.

Generalizations

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

- **SCE** *ElementType* (see the section **SCE** specification for more information).

Properties

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

Table 152. ElementRelationshipType Attributes and/or Associations

Property/Association	Description
relationshipKindRef : RelationshipKind [1]	A description of the type of the relationship. See *RelationshipKind*, below, for more details.
sourceMultiplicity : String [0..1]	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.
sourceRef : ElementType [1]	The source concrete specialization of *ElementType* of the relationship.

targetMulitplicity : 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.

12.4.3 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).

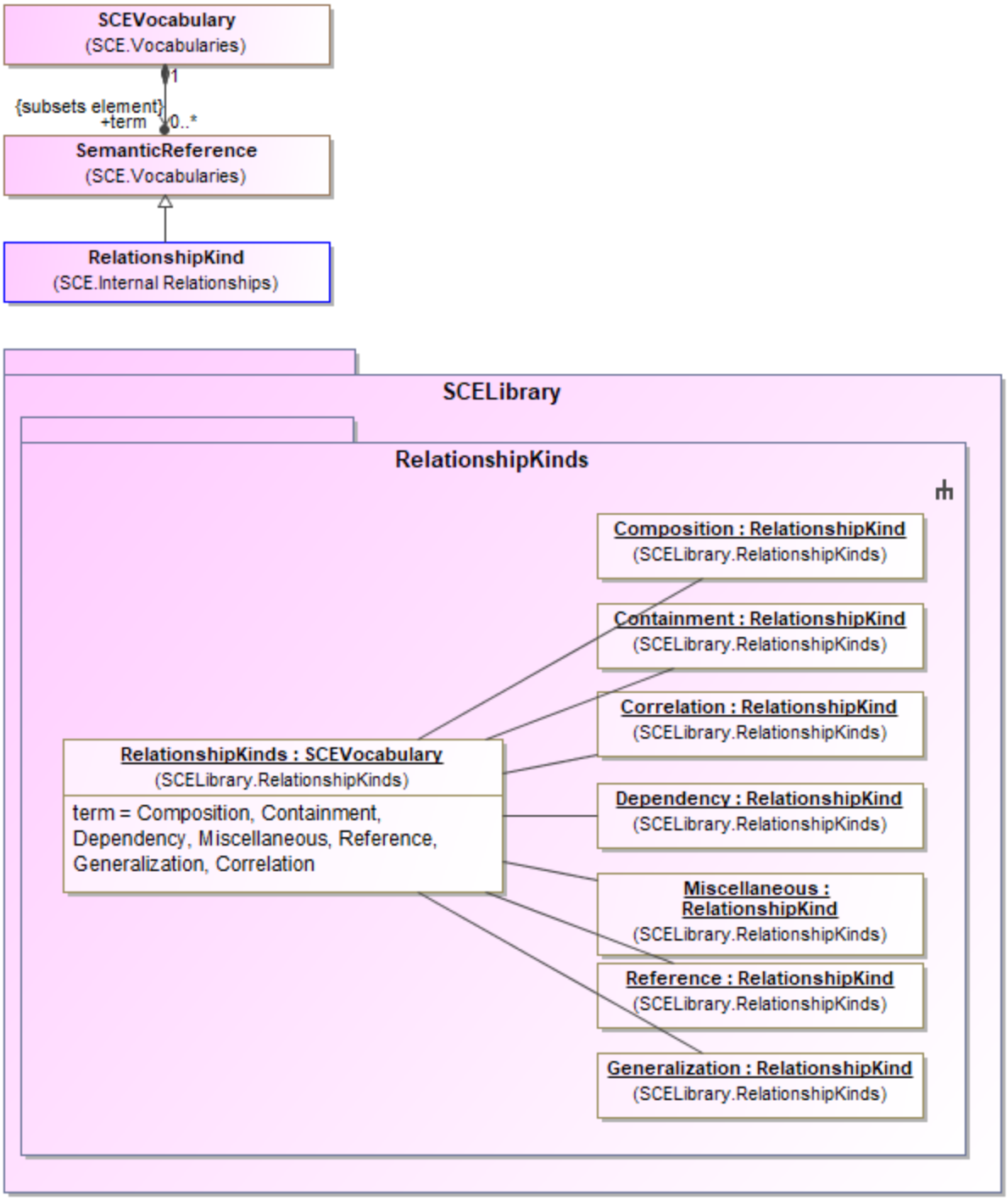


Figure 89: RelationshipKind MM

Generalizations

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

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

Properties

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

12.5 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.

12.5.1 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 *SCEDefinitions* or a sub-package *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 connectorconnection 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.

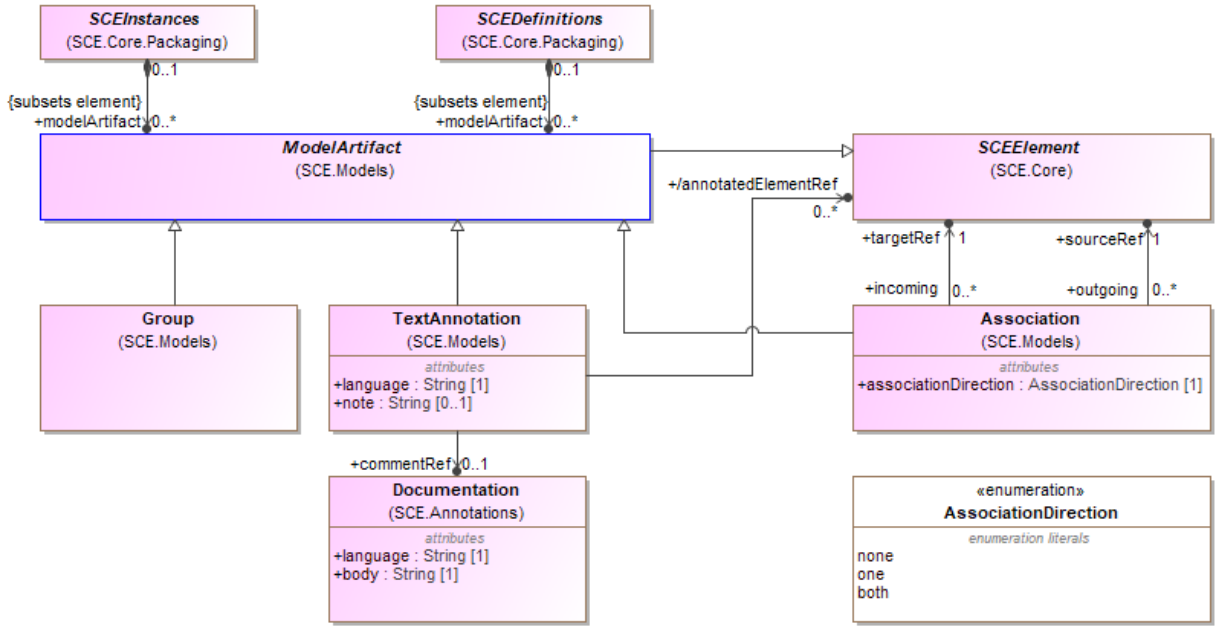


Figure 90: The ModelArtifact Metamodel

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

12.5.2 Association

Figure 91: An Association

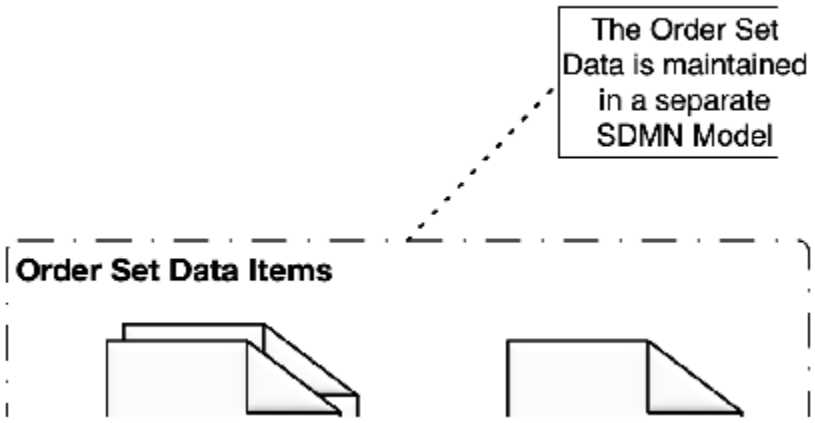


Figure 92: An Association Used with a Text Annotation

Generalizations

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

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

Properties

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

Table 153. Association Attributes and/or Associations

Property/Association	Description
associationDirection : AssociationDirection [1]	`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.
sourceRef : SCEElement [1]	The *SCEElement* that the **Association** is connecting from.

targetRef : SCEElement [1]	The *SCEElement* that the **Association** is connecting to.

12.5.3 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 154. AssociationDirection Literals

Literal	Description
both	A value of "both" means that there SHALL be an arrowhead at both ends of the **Association**.
none	The default is "none" (no arrowhead).
one	A value of "one" means that the arrowhead SHALL be at the *targetRef* Object.

12.5.4 Group

Figure 93: A Group

Generalizations

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

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

Properties

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

12.5.5 TextAnnotation

Figure 94: A Text Annotation

Generalizations

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

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

Properties

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

Table 155. TextAnnotation Attributes and/or Associations

Property/Association	Description
annotatedElementRef : SCEElement [0..*]	If the **TextAnnotation** is associated with (is the `source` of an **Association**) another model element, this association will identify the `target` of the **Association**. It is derived from the connected **Association** element.
commentRef : Documentation [0..1]	`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 *SCEPackage*. 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 used, then the `note` attribute SHALL NOT be used.
language : String [1]	The named `language` can be a natural language, in which case the `body` is an informal representation, or an artifical language, in which case the `body` is expected to be a formal, machine-parsable representation. If the `note` attribute is used, then the `language` attribute is required.

note : String [0..1]	`Note` is one of two attributes that provides text that the modeler wishes to communicate to the reader of the diagram. The text within a `note` is contained in and specific to the diagram where the **TextAnnotation** is placed. This attribute is optional, but if it used, then the `commentRef` attribute SHALL NOT be used.

12.5.6 Model 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*.

12.6 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 attributes and associations for the *SCEVocabulary* element:

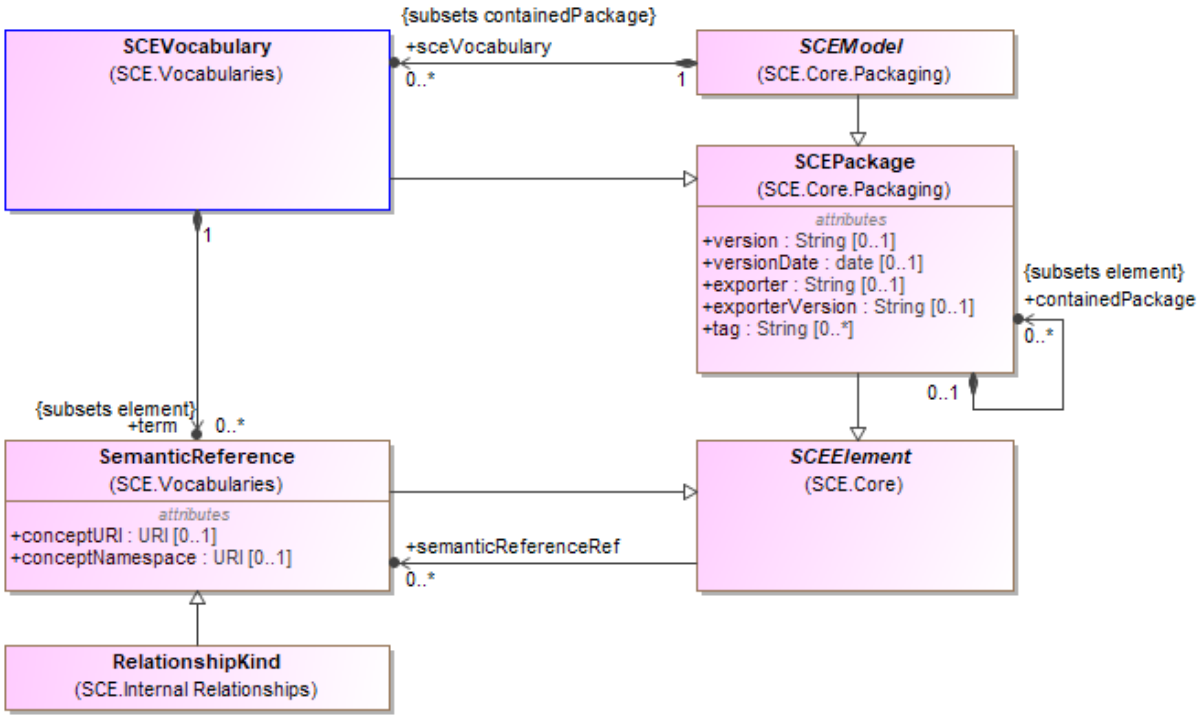


Figure 95: The SCEVocabulary Metamodel

12.6.1 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).

Properties

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

Table 156. SCEVocabulary Attributes and/or Associations

Property/Association	Description
term : SemanticReference [0..*]	A set of entries in the vocabulary that may have references to more detailed definitions of the term.

12.6.2 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.

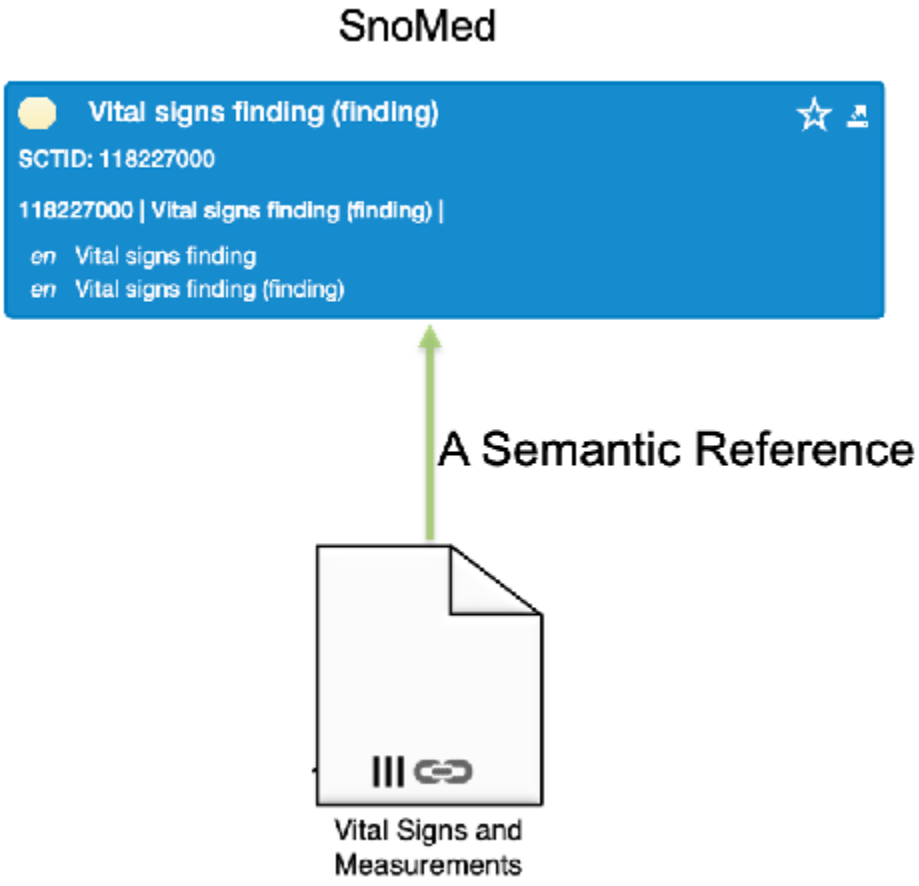


Figure 96: An Example of a Semantic Reference within a SDMN Model

Generalizations

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

- **SCE** *SCEElement* (see the section **SCE** specification for more information).

Properties

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

Table 157. SemanticReference Attributes and/or Associations

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

conceptURI : URI [0..1]	This attribute defines the URI location of the target of the *SemanticReference*.

13 SCELibrary

A Library is included in **SCE** to provide standard instances that are intended to 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).

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

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

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

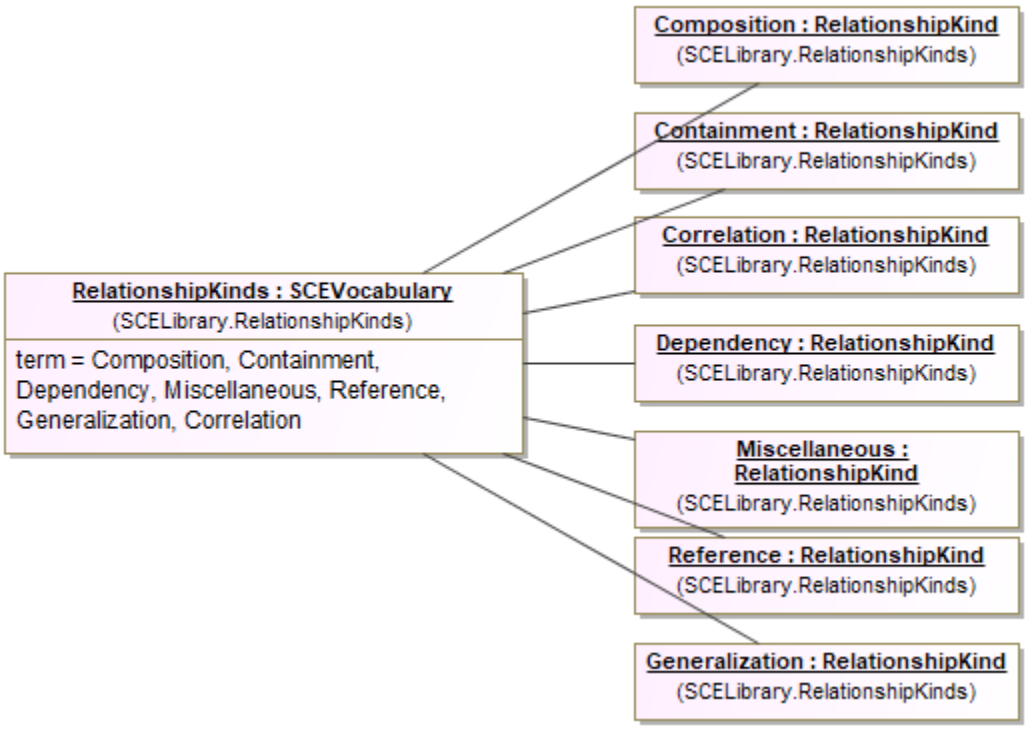


Figure 97: The RelationshipKinds Instance Model

13.1.1 Composition

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

13.1.2 Containment

`Containment` indicates that the `source` element is a container for the `target` element.

13.1.3 Correlation

`Correlation` indicates that the `source` element is correlated with the `target` element. This is often used when a mapping is required between the structures of two data elements.

13.1.4 Dependency

`Dependency` indicates that `target` element is dependent in some way on the `source` element.

13.1.5 Generalization

`Generalization` indicates that the `source` element is a generalization of the `target` element (which is based on and extends the `source`).

13.1.6 Miscellaneous

`Miscellaneous` indicates that `source` element has some relationship with the `target` element that is of a kind that is not expressed through the other *RelationshipKind* instances.

13.1.7 Reference

`Reference` indicates that `source` element references the `target` element.

14 PPMN and Parties Diagram Interchange (PPMN DI and Parties DI)

14.1 Scope

This chapter describes the **PPMN** and **Parties** Diagram Interchange (**PPMN DI** and **Parties DI**, respectively). **PPMN DI** extends the **Parties DI**. The **Parties DI** uses the diagram interchange capabilities provided in **SCE** (see the SCE 1.0 Beta 1 specification (dtc/22-01-04)). The **PPMN DI** is meant to facilitate the interchange of **PPMN** and **Parties** diagrams between tools rather than being used for internal diagram representation by the tools. The simplest interchange approach to ensure the unambiguous rendering of **PPMN** and **Parties** diagrams was chosen. As such, **PPMN DI** does not aim to preserve or interchange any “tool smarts” between the source and target tools (e.g., layout smarts, efficient styling, etc.).

PPMN DI does not ascertain that **PPMN** or **Parties** diagrams are syntactically or semantically correct.

14.2 Diagram Definition and Interchange

PPMN DI and **Parties DI**, through their extension of the **SCE DI** meta-model are defined as a MOF-based meta-models. As such, their instances can be serialized and interchanged using XMI. **PPMN DI** and **Parties DI** are also defined by the **SCEDI** XML schema. Thus, their instances can also be serialized and interchanged using XML.

The **SCE DI** (see the **SCE 1.0 Beta 1** specification) is 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 DI.

The focus of **PPMN DI** and **Parties DI** is the interchange of laid out shapes and edges that constitute **PPMN** and **Parties** diagrams, respectively. Each shape and edge references a particular **PPMN** or **Parties** model element. The referenced model elements are all part of an actual **PPMN** or **Parties** model. As such, **PPMN DI** and **Parties DI** are meant to only contain information that is neither present nor derivable, from the original model whenever possible. Simply put, to render a **PPMN** or **Parties** diagram both the proper **DI** instance(s) (including **PPMN**, **Parties, and SCE DI** instances**)** as well as the referenced **PPMN** and/or **Parties** model instance(s) are REQUIRED.

From the **PPMN DI** perspective, a **PPMN** diagram is a particular snapshot of a **PPMN** model at a certain point in time. Multiple **PPMN** diagrams can be exchanged referencing model elements from the same **PPMN** model. Each diagram may provide an incomplete or partial depiction of the content of the **PPMN** model. 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. Similarly for **Parties DI**.

14.3 Notation

As a specification that contains elements that can notated graphically, **PPMN** specifies the depiction for **PPMN** diagram elements, including **Parties** elements and **SCE** *DiagramArtifact* elements.

Serializing a **PPMN** diagram (including those that contain only **Parties** model elements) for interchange requires the specification of a collection of *SCEShape*(s) and *SCEEdge*(s) in the *SCEDiagram*. The *SCEShape*(s) and *SCEEdge*(s) attributes must be populated in such a way as to allow the unambiguous rendering of the **PPMN** diagram by the receiving party. More specifically, the *SCEShape*(s) and *SCEEdge*(s) MUST reference **PPMN (**or **Parties)** model elements. If no *SCEElement* is referenced or if the reference is invalid, it is expected that this shape or edge will not be depicted.

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

14.3.1 Labels

Both *SCEShape* and *SCEEdge* elements 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*.

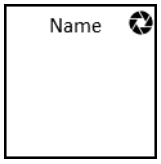
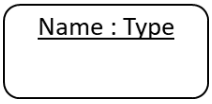
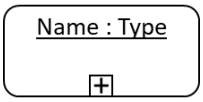
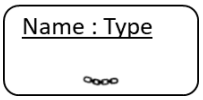
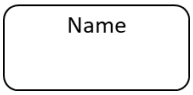
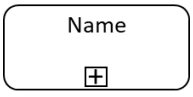
14.3.2 Shape Resolution

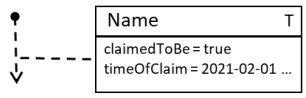
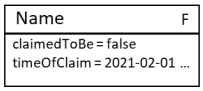
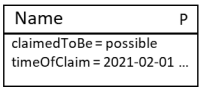
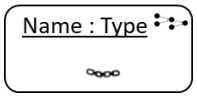
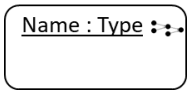
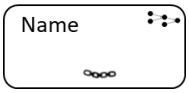
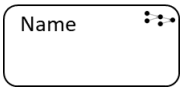
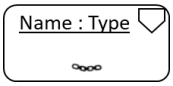
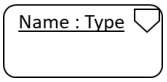
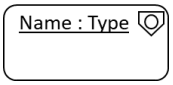
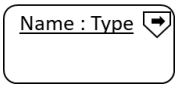
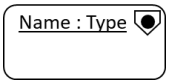
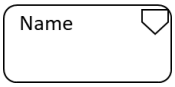
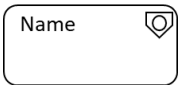
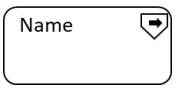
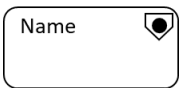
SCEShape can be used to represent any of the non-relationship elements from **PPMN** and **Parties** models. These include elements such as *Entity*, *EntityType*, *Occurrence*, *OccurrenceType*, *Organization*, and *OrganizationType.*When a *SCEShape* is used to depict a diagram element the actual shape is determined by the referred PPMN or Parties model element.

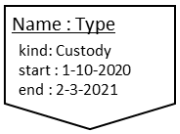
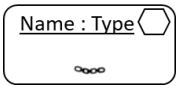
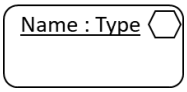
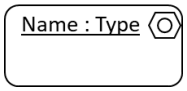
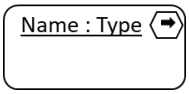
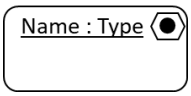
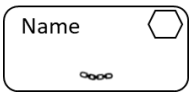
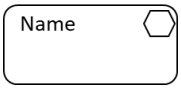
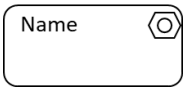
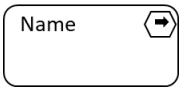
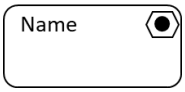
14.3.2.1 Depiction for PPMN Diagram Elements

The following table presents the depiction resolutions for **PPMN** elements:

Table 158. Depiction Resolution of PPMN Shapes

PPMN Element	PPMN Element Attributes	Depiction
Entity		
EntityType		
EntitySnapshot		
EntityTypeSnapshot		
EntityFormat		
Occurrence		
Occurrence (with Subchain)		
OccurrenceChain		
OccurrenceType		
OccurrenceChainType (with Subchain)		
OccurrenceBranchNode		

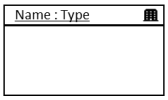
Claim (as shape)	claimedToBe = true	
Claim (as shape)	claimedToBe = false	
Claim (as shape)	claimedToBe = possible	
PedigreeChain		
PedigreeOccurrence		
PedigreeChainType		
PedigreeOccurrenceType		
CustodyChain		
CustodyOccurrence		
CustodyOccurrence (Custody Start)	kind = instance of CustodyStartKind	
CustodyOccurrence (Custody Transfer)	kind = instance of CustodyTransferKind	
CustodyOccurrence (Custody Start)	kind = instance of CustodyEndKind	
CustodyChainType		
CustodyOccurrenceType		
CustodyOccurrenceType (CustodyStart type)	kind = instance of CustodyStartKind	
CustodyOccurrenceType (CustodyTransfer type)	kind = instance of CustodyTransferKind	
CustodyOccurrenceType (CustodyEnd type)	kind = instance of CustodyEndKind	

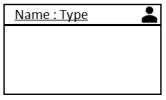
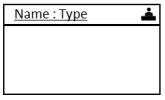
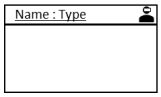
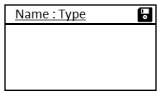
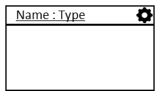
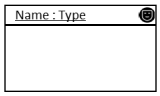
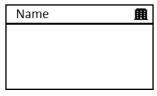
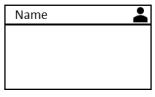
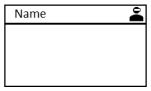
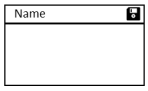
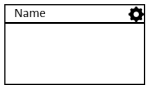
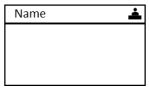
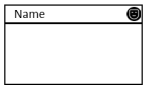
Custody (with attributes)		
OwnershipOccurrenceChain		
OwnershipChangeOccurrence		
OwnershipChangeOccurrence (Acquisition)	kind = instance of OwnershipStartKind	
OwnershipChangeOccurrence (Ownership Change)	kind = instance of OwnershipTransferKind	
OwnershipChangeOccurrence (End of Ownership Chain)	kind = instance of OwnershipEndKind	
OwnershipChainType		
OwnershipOccurrenceType		
OwnershipOccurrenceType (Ownership Start)	kind = instance of OwnershipStartKind	
OwnershipOccurrenceType (Ownership Transfer)	kind = instance of OwnershipTransferKind	
OwnershipOccurrenceType (ownership End)	kind = instance of OwnershipEndKind	
Ownership (with attributes)		

14.3.2.2 Depiction for Parties Diagram Elements

The following table presents the depiction resolutions for **Parties** elements:

Table 159. Depiction Resolution of Parties Shapes

Parties Element	Parties Element Attributes	Depiction
Organization		

Person		
Position		
NonHumanAgent		
Software		
Machinery		
PartyRole		
OrganizationType		
IndividualType (Person)	kind = Person	
IndividualType (NonHumanAgent)	kind = NonHumanAgent	
IndividualType (Software)	kind = Software	
IndividualType (Machinery)	kind = Machinery	
PositionType		
PartyRoleType		
Area		
Path		
PhysicalAddress		
NetworkAddress		
GeospacialExtent		

SpaceTime		

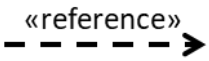
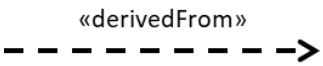
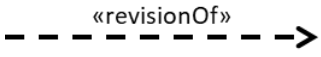
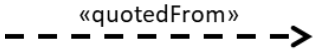
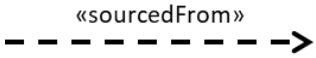
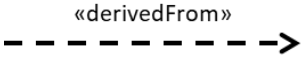
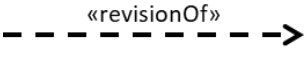
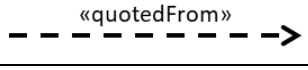
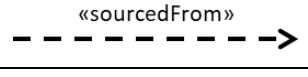
14.3.3 Edge Resolution

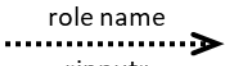
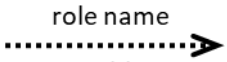
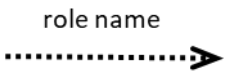
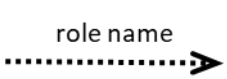
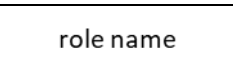
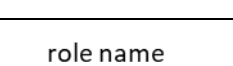
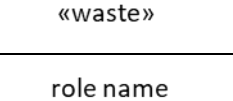
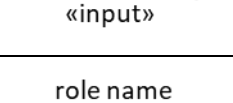
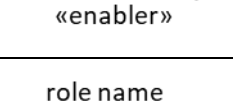
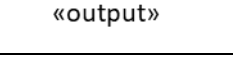
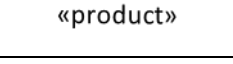
SCEEdge can be used to represent and of the **PPMN** or **Parties** relationships including relationships such as *EntityRelationship*, *OccurrenceDependency*, and *PartyRelationship*.

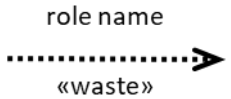
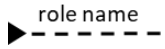
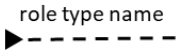
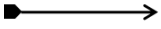
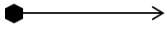
14.3.3.1 Depiction for PPMN Diagram Elements

The following table presents the depiction resolutions for **PPMN** edges:

Table 160. Depiction Resolution of PPMN Edges

PPMN Element	PPMN Element Attribute	Depiction
EntityRelationship (Generalization)	relationshipKindRef = Generalization	
EntityRelationship (Containment)	relationshipKindRef = Containment	
EntityRelationship (Composition)	relationshipKindRef = Composition	
EntityRelationship (Dependency)	relationshipKindRef = Dependency	
EntityRelationship (Miscellaneous)	relationshipKindRef = Miscellaneous	
EntityRelationship (Reference)	relationshipKindRef = Reference	
DerivedFrom		
RevisionOf		
QuotedFrom		
SourcedFrom		
DerivationType (DerivedFrom)	kind = DerivedFrom	
DerivationType (RevisionOf)	kind = RevisionOf	
DerivationType (QuotedFrom)	kind = QuotedFrom	
DerivationType (SourcedFrom)	kind = SourcedFrom	

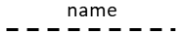
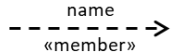
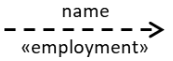
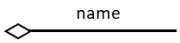
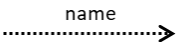
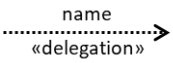
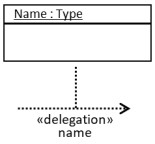
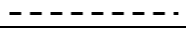
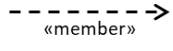
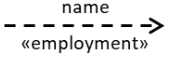
OccurrenceRelationship		
OccurrenceDependency	kind = Input	
OccurrenceDependency	kind = Enabler	
OccurrenceDependency	kind = Output	
OccurrenceDependency	kind = Product	
OccurrenceDependency	kind = By-product	
OccurrenceDependency	kind = Waste	
OccurrenceDependencyType	kind = Input	
OccurrenceDependencyType	kind = Enabler	
OccurrenceDependencyType	kind = Output	
OccurrenceDependencyType	kind = Product	
OccurrenceDependencyType	kind = By-product	

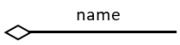
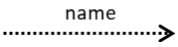
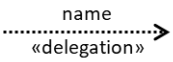
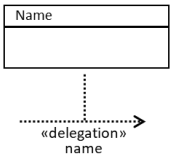
OccurrenceDependencyType	kind = Waste	
OccurrenceRole		
OccurrenceRoleType		
OccurrenceGraphTransition	relationshipKind = Transition	
Custody (as relationship)		
CustodyType (as relationship)		
Ownership (as relationship)		
OwnershipType (as relationship)		

14.3.3.2 Depiction for Parties Diagram Elements

The following table presents the depiction resolutions for **Parties** edges:

Table 161. Depiction Resolution of Parties Edges

Parties Element	Parties Element Attribute	Depiction
PartyRelationship (General)	relationshipKind = General	
PartyRelationship (Member)	relationshipKind = Member	
PartyRelationship (Employment)	relationshipKind = Employment	
OrganizationalStructureRelationship	relationshipKind = Part	
PositionAssignment	relationshipKind = PositionAssignment	
Delegation (without Authority shown)		
Delegation (with Authority shown)	authority = *not null*	
PartyRelationshipType (General)	relationshipKind = Member	
PartyRelationshipType (Member)	relationshipKind = Member	
PartyRelationshipType (Employment)	relationshipKind = Employment	

PartyRelationshipType (Part)	relationshipKind = Part	
PositionAssignmentType	relationshipKind = PositionAssignment	
DelegationType (without Authority shown)	relationshipKind = Delegation	
DelegationType (with Authority shown)	relationshipKind = Delegation	

Annex A: PROV Traceability

(informative)

A key requirement of **PPMN** is to support all the capabilities available in the W3C PROV specification. This ANNEX describes the traceability of PPMN elements to elements in W3C PROV. Please note that the model of the W3C PROV specification presented herein is an interpretation in UML of that specification by the PPMN authors.

This diagram shows the PPMN and W3C PROV concepts related to the primary three PROV elements - Agent, Entity, and Activity.

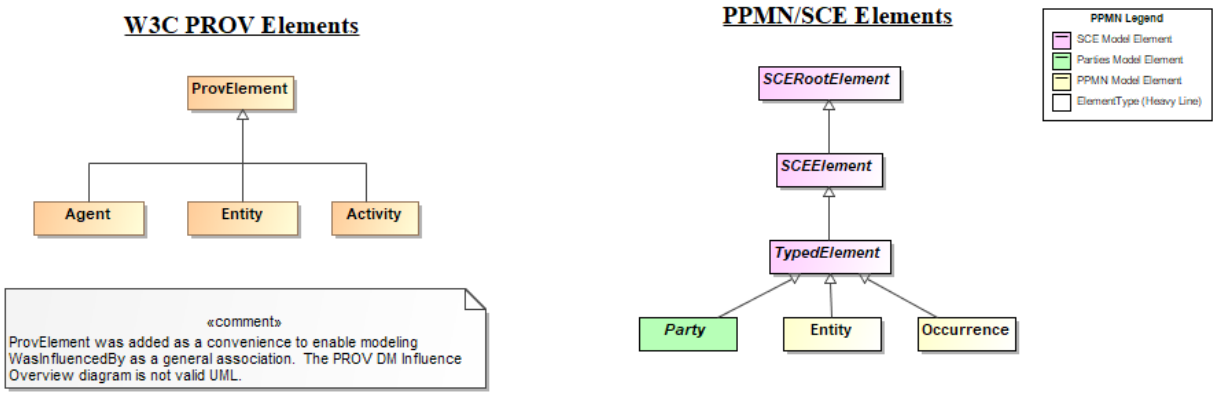


Figure 98: PPMN Trace to PROV - Primary PROV Elements

This diagram shows the PPMN and W3C PROV concepts related to Agents, Responsibility, and Influence.

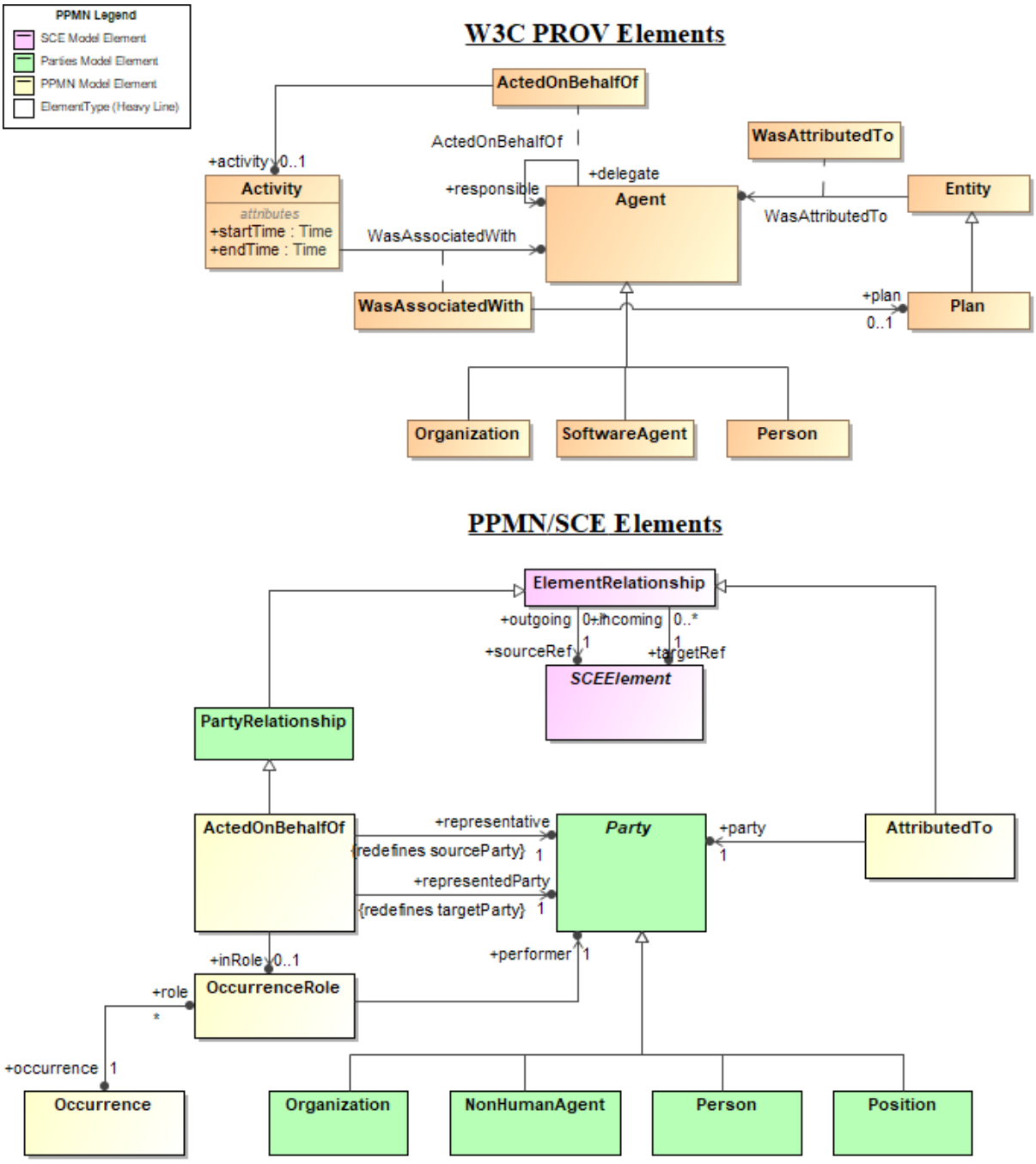


Figure 99: PPMN Trace to PROV - Agents, Responsibility, and Influence

This diagram shows the PPMN and W3C PROV concepts related to Derivations.

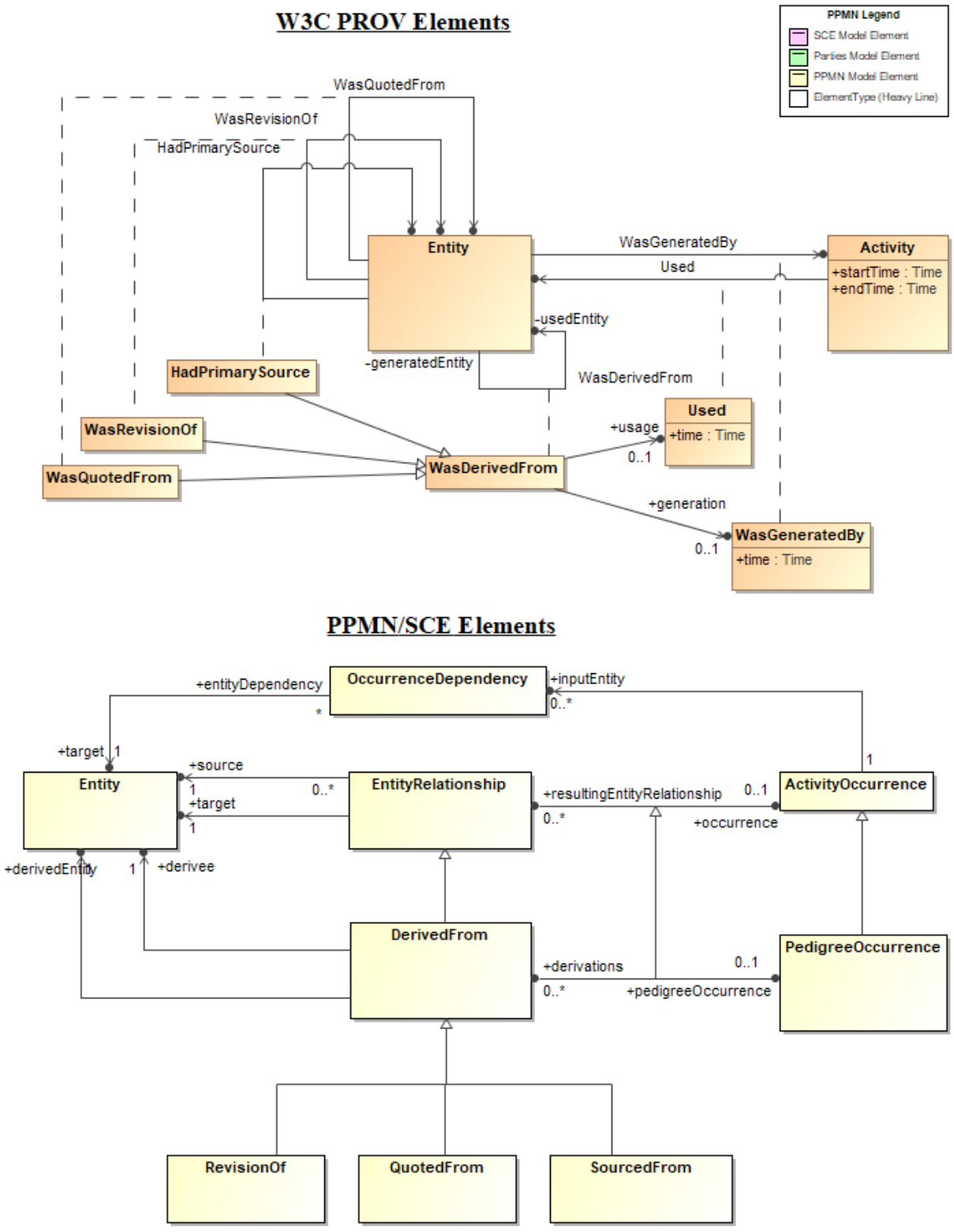


Figure 100: PPMN Trace to PROV - Derivations

This diagram shows the PPMN and W3C PROV concepts related to Entities and their relationships to Activities (or Occurrences in PPMN).

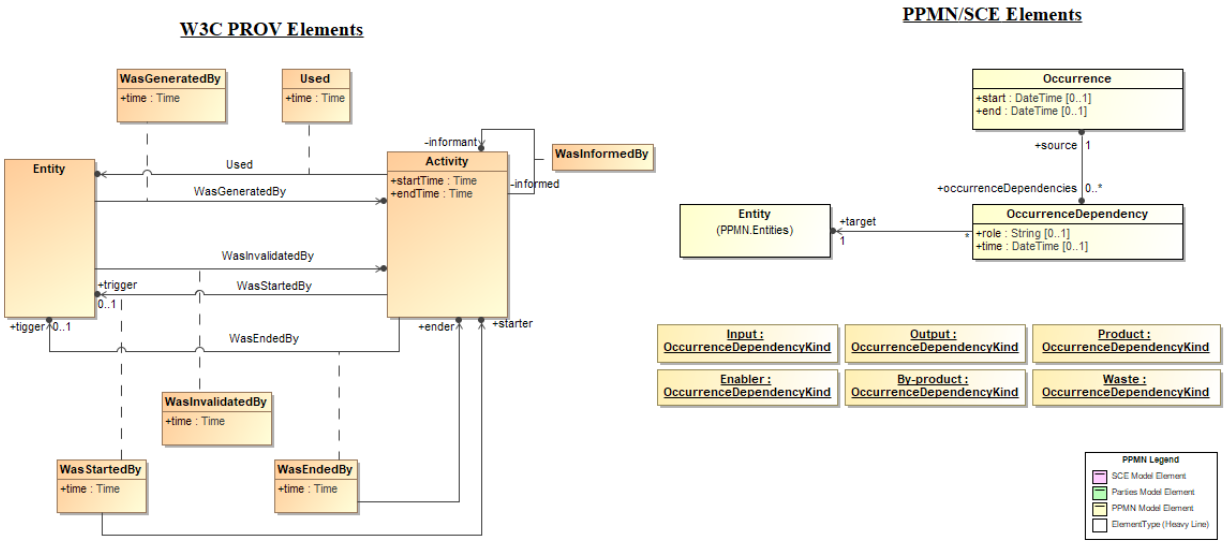


Figure 101: PPMN Trace to PROV - Entities and Activities

This diagram shows the PPMN and W3C PROV concepts related to Influence.

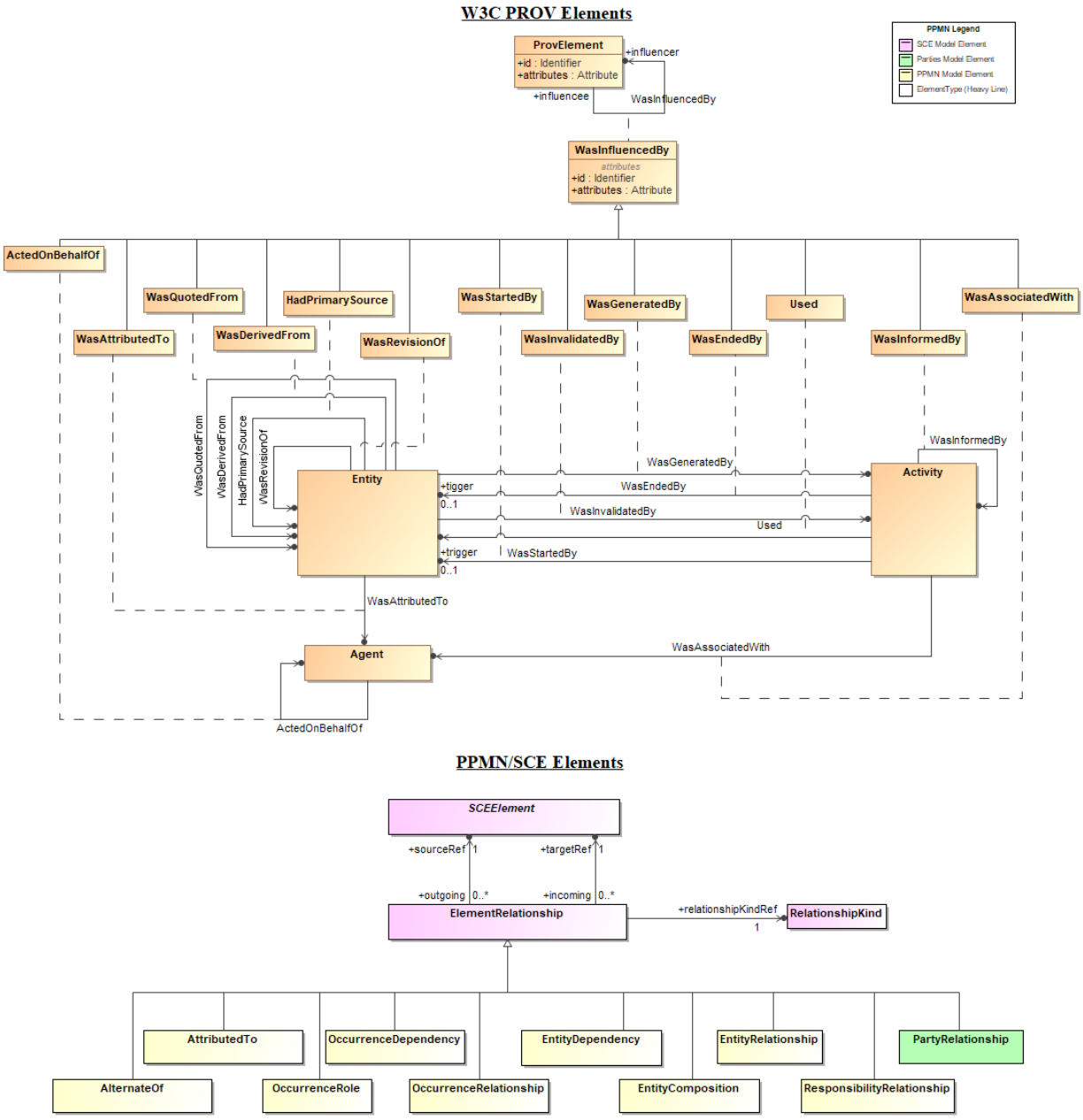


Figure 102: PPMN Trace to PROV - Influence

This diagram shows the PPMN and W3C PROV concepts related to the core PROV elements.

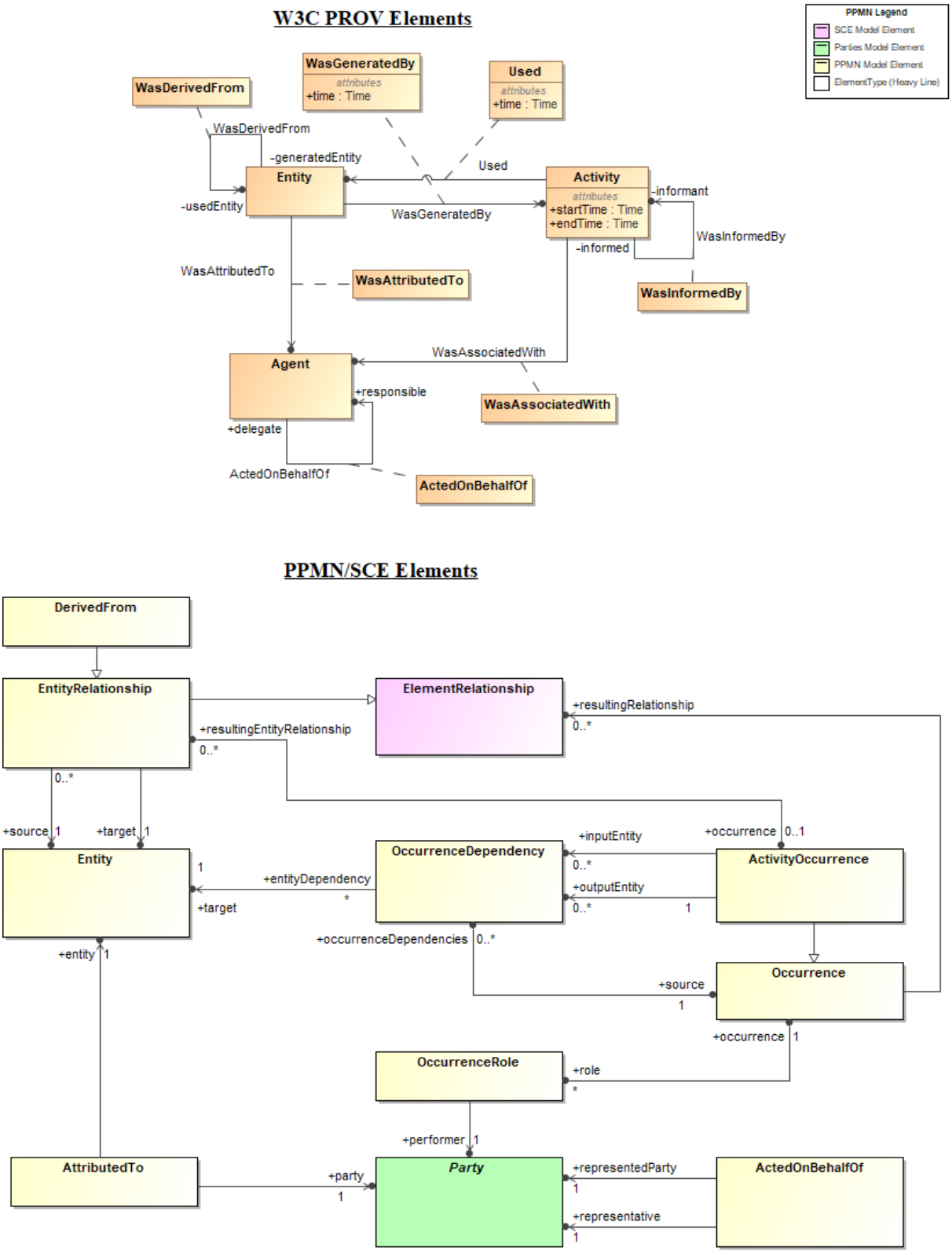
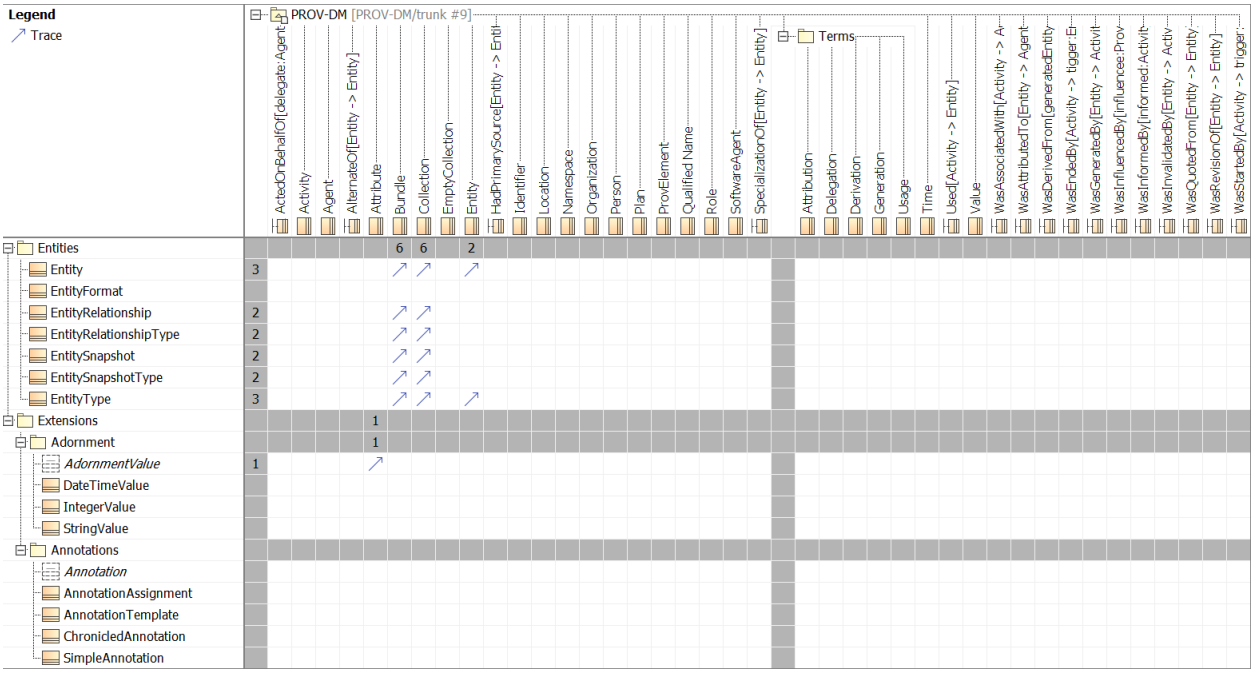


Figure 103: PPMN Trace to PROV - PROV Core Structures

This traceability matrix shows the traceability of PPMN elements to W3C PROV elements related to Elements,

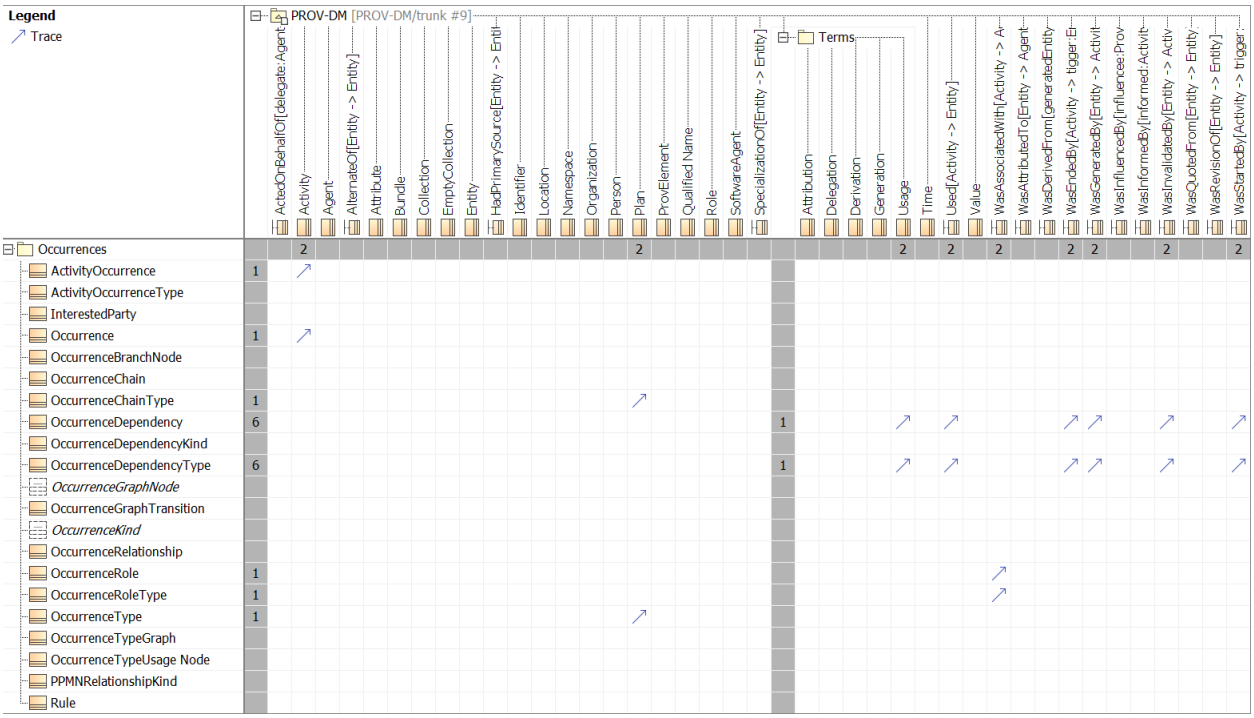
Entities, and Extensions.

Table 162. PPMN to PROV Traceability Matrix - Elements, Entities and Extensions



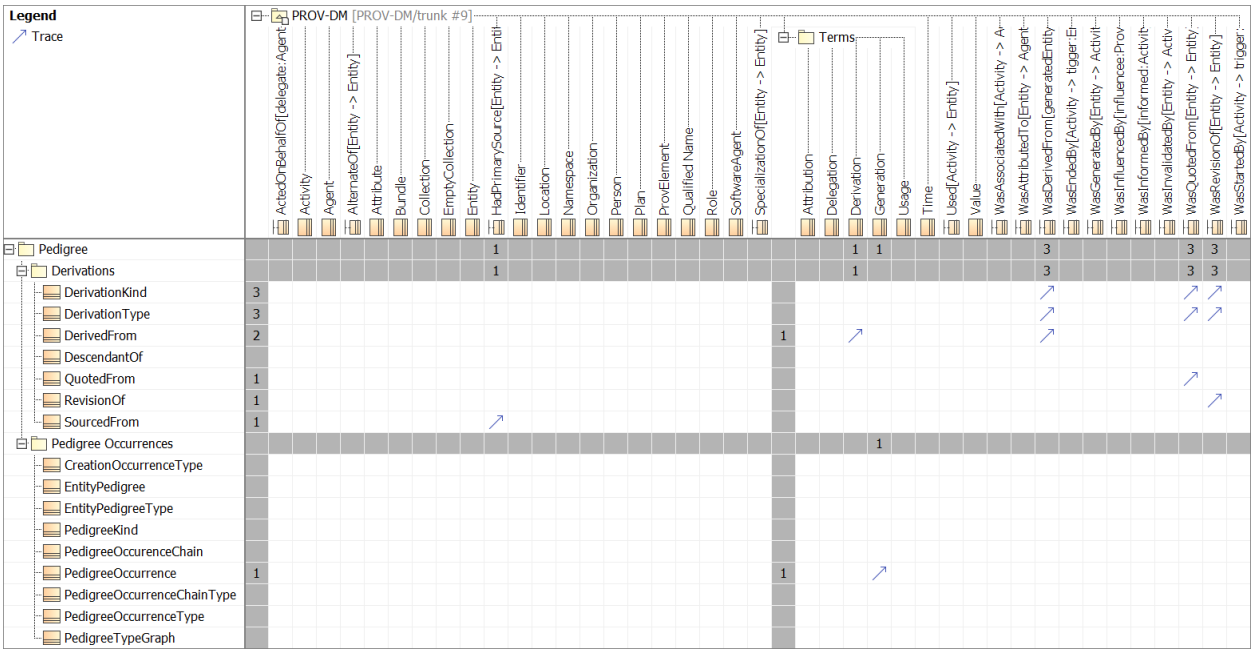
This traceability matrix shows the traceability of PPMN elements to W3C PROV elements related to Activities.

Table 163. PPMN to PROV Traceability Matrix - Activities



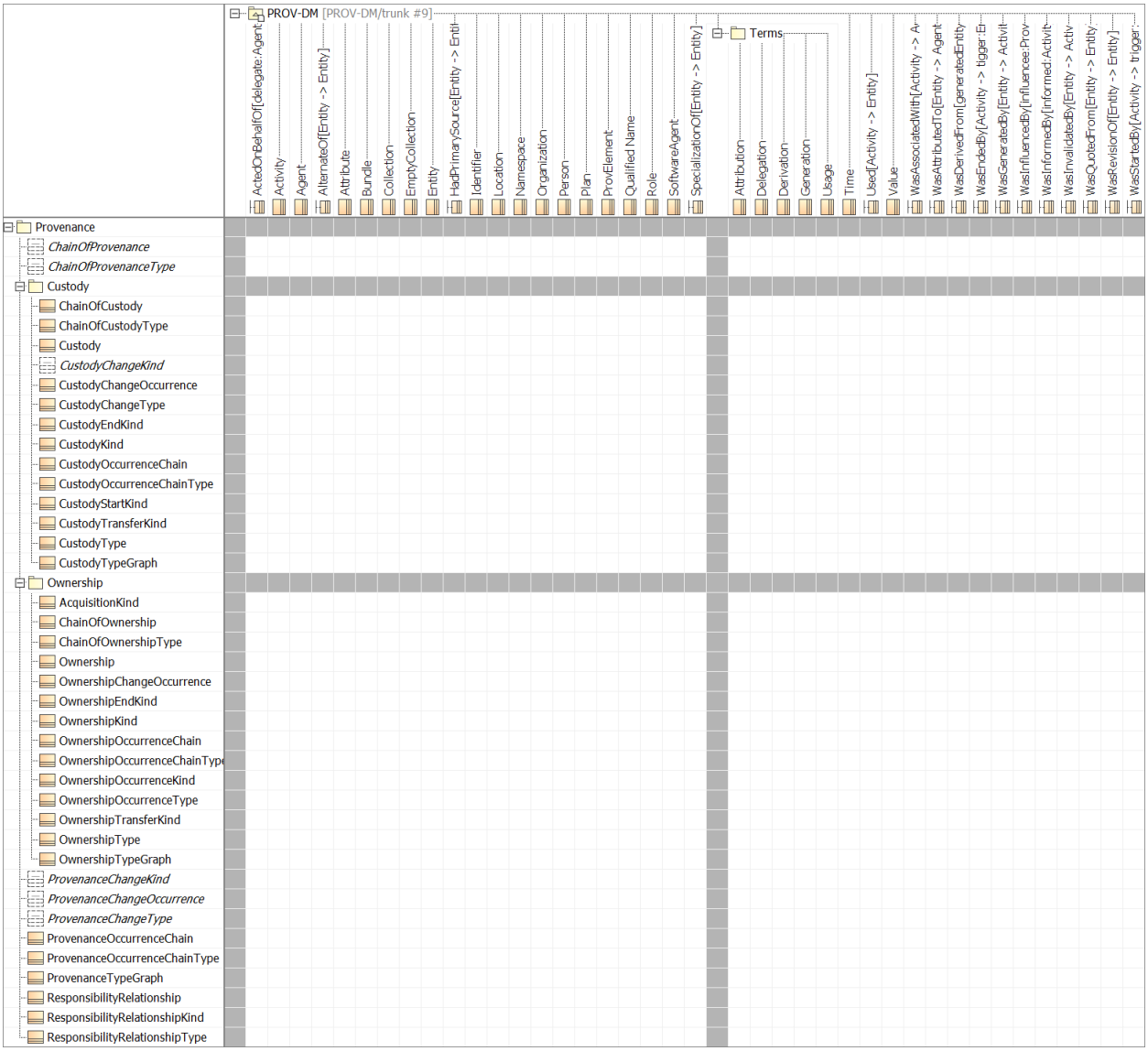
This traceability matrix shows the traceability of PPMN elements to W3C PROV elements related to Pedigree.

Table 164. PPMN to PROV Traceability Matrix - Pedigree



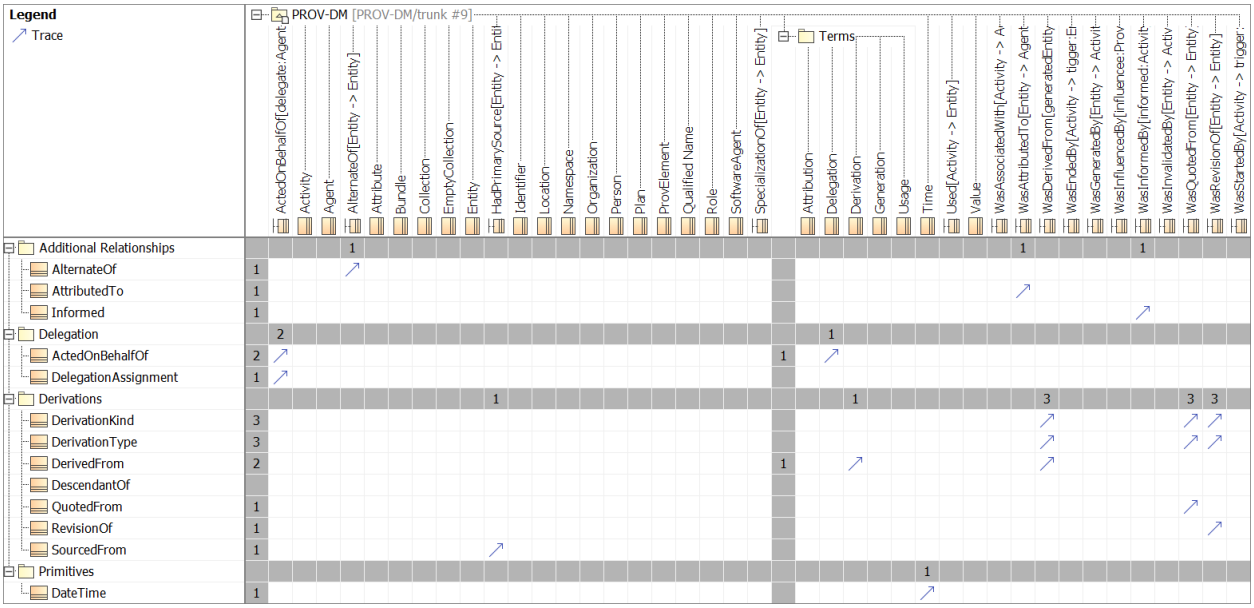
This traceability matrix shows the traceability of PPMN elements to W3C PROV elements related to Provenance.

Table 165. PPMN to PROV Traceability Matrix - Provenance



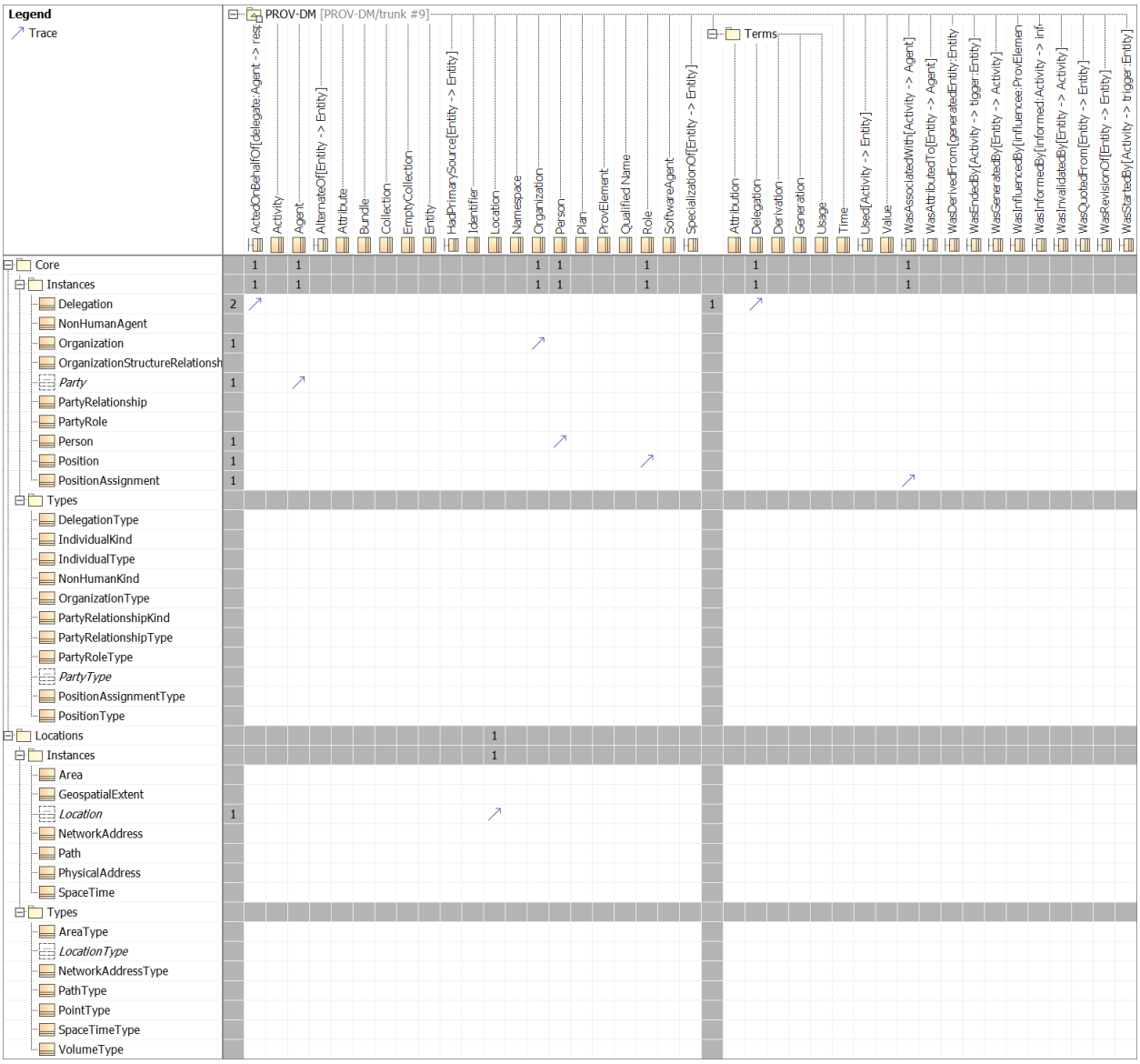
This traceability matrix shows the traceability of PPMN elements to W3C PROV elements related to Delegation, Derivation, Primitives and other relationships.

Table 166. PPMN to PROV Traceability Matrix - Delegations, Derivations, Primitives and Other Relationships



This traceability matrix shows the traceability of PPMN elements to W3C PROV elements related to Parties and Locations.

Table 167. PPMN to PROV Traceability Matrix - Parties and Locations



TBD

Table 168. SCE to PROV Traceability Matrix