# Product Lifecycle Management Specification

This OMG document replaces the submission (mantis/07-03-01) and the draft adopted specification (dtc/07-04-01). It is an OMG Final Adopted Specification, which has been approved by the OMG board and technical plenaries, and is currently in the finalization phase. Comments on the content of this document are welcomed, and should be directed to *issues@omg.org* by November 19, 2007.

You may view the pending issues for this specification from the OMG revision issues web page *http://www.omg.org/issues/*; however, at the time of this writing there were no pending issues.

The FTF Recommendation and Report for this specification will be published on March 21, 2008. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

**OMG Adopted Specification**
**dtc/07-05-01**

**Date**: May 2007

# Product Lifecycle Management Services, V2.0

Final Adopted Specification
dtc/07-05-01

## USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

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

## LICENSES

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

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

## PATENTS

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

## GENERAL USE RESTRICTIONS



## DISCLAIMER OF WARRANTY

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

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

## RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

## TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ and OMG Interface Definition Language (IDL)™ are trademarks of the Object Management Group. 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 *http://www.omg.org*, under Documents, Report a Bug/Issue (*http://www.omg.org/technology/agreement.htm*).

# Table of Contents

# Preface

## About the Object Management Group

### 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 *http://www.omg.org/*.

## OMG Specifications

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

*http://www.omg.org/technology/documents/spec_catalog.htm*

Specifications within the Catalog are organized by the following categories:

### OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications.

### OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM).

### Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications.

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
140 Kendrick Street
Building A, Suite 300
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 *http://www.iso.org*

## Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

**Helvetica/Arial - 10 pt. Bold:** OMG Interface Definition Language (OMG IDL) and syntax elements.

**`Courier - 10 pt. Bold:`** Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

**Note –** Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

## Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*.

# 1 Scope

This specification defines a Platform Independent Model (PIM) for Product Lifecycle Management Services V2.0. Its informational model is derived from the ISO 10303-214 STEP model by an EXPRESS-X mapping specification and a EXPRESS-to-XMI mapping process. The functional model is derived from the OMG PDM Enablers V1.3 and to fulfill requirements of the RFP.

The specification defines a Platform Specific Model (PSM) applicable to the Web Services implementation defined by a WSDL specification, with a SOAP Binding, and an XML Schema specification.

# 2 Conformance

An implementation compliant to this specification shall support the XML Schema and Web Services PSM described in this specification and shall be capable to deliver and to consume valid XML documents with respect to the XML Schema defined in the PSM of this specification via service interfaces defined in WSDL in the PSM of this specification.

An implementation compliant to the XML Schema and Web Services PSM described in this specification shall support at least one of the Queries Conformance Points listed below.

A Queries Conformance Point consists of a set of specializations of the type `Query`. This specification defines five Queries Conformance Points:

1. the Generic Queries Conformance Point (see Section 9.4 ),
2. the XPath Queries Conformance Point (see Section 9.5 ),
3. the Specific Queries Conformance Point (see Section 9.7 ),
4. the PDTnet Queries Conformance Point (see Section 9.8 ), and
5. the Messages Queries Conformance Point (see Section 9.9 ).

An implementation shall define the Queries Conformance Points it is realizing.

The Conformance Points are independent from each other.

# 3 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.

- UML Infrastructure Specification, v2.1.1; OMG document formal/07-02-04
- UML Superstructure Specification, v2.1.1; OMG Document formal/07-02-03
- Meta Object Facility (MOF) Core, v2.0; OMG Document formal/06-01-01
- Meta Object Facility (MOF) 2.0 XMI Mapping Specification, v2.1; OMG Document formal/05-09-01
- ISO 10303-11:1994 Description methods: The EXPRESS language reference manual
- ISO 10303-14:2001 Description methods: The EXPRESS-X language reference manual

- ISO CD 10303-25:2003 Implementation methods: EXPRESS to UML mapping
- ISO TS 10303-28:2002 XML representation for EXPRESS-driven data
- ISO 10303-203:2000 Configuration-controlled mechanical design
- ISO 10303-214:2000 Core data for automotive mechanical design process
- ISO 10303-232:2001 Technical Data Package
- ISO/IEC 10746: Reference Model for Object Distributed Computing (RM/ODP)

# 4 Terms and Definitions

The following terms are defined in reference specifications which are used in this specification.

Conformance Class: defined in ISO 10303 (STEP) Part 1 as a subset of the informational model defined in Application Protocol (AP) for which conformance may be claimed.

Data Exchange Tool: Application to realize EDI based on a choice of appropriate and agreed protocols.

# 5 Symbols

The following abbreviations from references specifications used in this specification.

**AIM**: Application Interpreted Model, defined in ISO 10303 (STEP) Part 1 as an information model that uses the integrated resources necessary to satisfy the information requirements and constraints of an application reference model (ARM), within an application protocol (AP)

**ARM**: Application Resource Model, defined in ISO 10303 (STEP) Part 1 as an information model that describes the information requirements and constraints of a specifc application context.

**AP**: Application Protocol, defined in ISO 10303 (STEP) Part 1 as a part of this International Standard that specifies an application interpreted model satisfying the scope and information requirements for a specifc application

**CC**: Conformance Class, see terms defined in Section 4

**CC21**: Conformance Class 21 is defined in ISO 10303 (STEP) AP214, 3$^{rd}$ edition, comprising information on product structure and configuration management data.

**ECM**: Engineering Change Management

**ECO**: Engineering Change Order

**ECR**: Engineering Change Request

**EDI**: electronic data interchange, automatic, one-way, asynchronous send of structured data between enterprises and application systems like ERP or PPS

**ENGDAT**: Engineering Data Message, industry standard for EDI targeted at the exchange of CAD data between multiple enterprises. ENGDAT is defined in the Odette standard ODG11ED9206 and as recommendation 4951 of VDA. It defines a structured meta data format containing sender and receiver information, purpose, status and approval data. ENGDAT is executed on top of the transport protocol OFTP.

**ERP**: Enterprise resource planning application systems integrates enterprise-wide information of an organization and support processes of that organization. Examples of modules in an ERP which formerly would have been stand-alone applications include: Manufacturing, Supply Chain, Financials, Customer Relationship Management, Human Resources, and Warehouse Management.

**Odette**: Organization for Data exchange by Tele Transmission in Europe. UK based non-profit organization of automotive manufacturers and suppliers standardizing on logistics, EDI and exchange of design data.

**OEM**:

**OFTP**: Odette File Transport Protocol corresponds to recommendation 4914/2 of VDA implemented on top of ISDN, X.25 or TCP/IP transport protocols.

**PDM**: Product data management is a concept to define, represent, and present data and documents related to the product development process. It implements processes and manages status changes of the product information.

**PDTnet**:

**PLM**: Product Lifecycle Management comprises all aspects of data management across the life cycle of a product and integrates them in a unified data base. Ideally all departments and systems related to the product refer to this central data base including the PPS and ERP systems, computer aided design (CAD), computer aided engineering (CAE), computer aided manufacturing (CAM) but also controlling, accounting, after-sales, customer care and service.

**PPS**: Production planning systems support resource planning and control and the related information management. They can be seen as a subset of ERP systems which usually extend PPS by human resource and finance management facilities.

**STEP**: Standard for the Exchange of product model data ISO 10303 is an international standard for the computer-interpretable representation and exchange of product data. ISO10303 is organized as a series of parts, each published separately.

**STEP PDM file**: A data exchange file formatted according to STEP 10303 Part 21 and storing information related to PDM data. The content of such a file is usually compliant to one of the Conformance Classes of a STEP AP.

**VDA** (in German: Verband der Automobilindustrie) German automotive industry association. Member of the European association Odette.

# 6 Additional Information

## 6.1 Changes to Adopted OMG Specifications

This specification completely replaces the PLM Services 1.0 specification. It is recommended that “PLM Services Version 1.0” is retired as an adopted technology.

## 6.2 How to Read this Specification

The rest of this document contains the technical content of this specification.

Although the chapters are organized in a logical manner and can be read sequentially, this is a reference specification is intended to be read in a non-sequential manner. Consequently, extensive cross-references are provided to facilitate browsing and search.

## 6.3 Accompanying Documents

The machine readable files accompanying this specification are provided in an extra structured archive document. The nature, the format, and a brief description of the provided files are given in Table 1.

**Table 1 - Accompanying documents**

| Filename | Nature | Format | Description |
|---|---|---|---|
| **express/** | | | |
| ./AIM2PIMEquivalence.exx | informative | STEP EXPRESS-X | Mapping of the relevant subset of AIM representation of STEP AP214 CC21 |
| ./PIM_Equivalence_ Schema.exp | informative | STEP EXPRESS | ARM representation of STEP AP214 CC21 model modified to be mapped to the UML PIM |
| **xmi** | | | |
| ./PIM_Informational Viewpoint.xmi | normative | XMI 2.1 | Platform independent infor-mational model of PLM Services 2.0 |
| ./PIM_Computational Viewpoint.xmi | normative | XMI 2.1 | Platform independent computational model of PLM Services 2.0 |
| ./PSM_XSDSchema.xmi | normative | XMI 2.1 | XML Schema Platform specific model of PLM Services 2.0 |
| ./PSM_WSDL namespace.xmi | normative | XMI 2.1 | WSDL Platform specific computational model of PLM Services 2.0 |
| **webservices/** | | | |
| ./xml | normative | XML Schema | XML Schema representation |
| ./wsdl | normative | WSDL | WSDL representation |

## 6.4 Informative References

The PLM V2.0 Specification refers to the following material

1. [WS-SECURITY] WS-Security Core Specification 1.1 OASIS
2. [TOKEN] Web Services Security Username Token Profile 1.1
3. [SEC] Web Services Security: SOAP Message Security 1.1
4. [ENC]XML-Encryption specification at the W3C organisation on http://www.w3.org/Encryption/2001/
5. [SIGNATURE] XML- Signature specification at the W3C organisation on http://www.w3.org/TR/xmldsig-core/
6. [VDA] Recommendation for Engineering Change Management Part 1: Engineering Change request (ECR), VDA recommendation 4965, VDA 2006.
7. [OASIS]http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
8. [PDMSchema]

9. [PDTnet] Project Information, available at http://www.prostep.org/en/infopool/pdtnet.

10. [SOAP] SOAP Version 1.2, June 2003, available at http://www.w3.org/TR/2003/REC-soap12-part[012]-20030624

11. [WSDL] Web Services Description Language (WSDL) 1.1, March 2001, available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315

12. [WS-I] WebServices interoperability Basic Profile version 1.1, http://www.ws-i.org/Profiles/BasicProfile-1.1.html

13. [XPATH] XML Path Language (XPath) 1.0, November 1999, available at http://www.w3.org/TR/1999/REC-xpath-19991116

## 6.5 Acknowledgements

The PLM V2.0 Specification contained in this document is the result of a joint effort from the following organizations:

- DaimlerChrysler AG
- avanion
- Robert Bosch GmbH
- Contact GmbH
- IBM
- IN GmbH
- Magna Steyr
- PartMaster GmbH
- PDTec GmbH
- proficiency
- PROSTEP AG
- SAP AG
- T-Systems International GmbH
- UGS
- valtech
- Volkswagen AG
- 88solutions.

# 7 Use Cases (informative)

This section describes the use cases supported by the specification. It utilizes use case, state machine and sequence diagrams as informative illustrations.

In general, use case diagrams are chosen if the process does not include control structures like alternatives, the relation of the single process steps is informal and each individual process can be seen as an isolated asset. State machine diagrams are used to express more formally specified control flows including transition states. Usually, not all of the process steps are useful without that context. Sequence diagrams are used if it is necessary or useful to illustrate a more detailed interaction of identified participants and systems of the use case.

## 7.1 Overview

The Information Model of the PLM Service is based on the STEP PDM Schema [PDMSchema] and extended by relevant subsets of STEP ISO 10303-214:2000, especially the Configuration Management modeling parts. The scope is chosen such that the informational model covers the complete Conformance Class CC21.

The selected scope of the Information Model is a superset of the model necessary to fulfill the requirement analysis in the PDTnet project [PDTnet]. The use cases identified in this industrial project of European automotive companies are given in brief in Section 7.2. The information model represents the PLM reference model in the informational viewpoint and is described in Chapter 8.

## 7.2 Detailed Use cases

This section describes the uses cases that are subject to the PLM services specification. They are categorized according to the requirement analysis resulting from the PDTnet project [PDTnet]. They are documented in this section, and may be extended continuously.

The scope of the use cases is defined supporting an online PLM integration scenario which is characterized by a data access on remote systems using internet functionality and technology. This integration does not provide a real online integration, but due to the usage of data streaming techniques and due to the possibility of an immediate reply by a system it comes near to it. It is assumed, that a neutral PLM client provides access to different PLM data providers (these are usually different PLM systems in different companies).

### 7.2.1 Export of assembly data

Export of product data (meta data and geometry) of assemblies and parts from one partner to another partner via exchange of ENGDAT packages (STEP PDM files, CAD files).

#### 7.2.1.1 Owner of the use case

This use case was defined by Work Group 1 of the PDTnet project.

#### 7.2.1.2 Process purpose

Export of product data which consist of meta data and geometry information of assemblies and its components from one partner to another partner via exchange of ENGDAT formatted packages. The ENGDAT message contains the STEP PDM files and (optionally) the CAD files, in native or neutral format.

#### 7.2.1.3 Partner role descriptions

**Table 7.1 - Roles for export of assembly data**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that selects and processes product data to be exported. | Person |
| PLM System | Party, that provides the relevant product data and functionality for product data management. This is usually a company's PLM system, which also can be extended by a tool, that provides extended STEP processor functionality. | System |
| Data Exchange (DE) Tool | System, that provides communication with a network and functionality to automatically process and pack/unpack file packages (usually ENGDAT-based). | System |

#### 7.2.1.4 Process definition

The process steps are:

1. User selects parts/documents/CAD models (using the functionality of the PLM system):
   - Selection of root/top level assembly by assembly (version) number
   - Selection of affected sub-assemblies or parts (could be controlled by a context or specific algorithm)
   - Exclusion of elements from selected set is possible
2. PLM system generates STEP PDM file:
   - Passing assembly structure tree and collecting transformation matrices (if appropriate)
   - Generating STEP PDM file
3. User selects addressee of data (using the DE tool or PLM tool)
4. Download of digital files from PLM system
5. DE Tool generates ENGDAT package including message abstract, STEP PDM file(s) and digital files (CAD/CC2 files, etc.)
6. DE Tool initiates sending of ENGDAT message

The order of the process steps could differ depending on specific user requirements and system scenario. Examples for possible alternative process step orders are:

a): 1. ® 3. ® 4. ® 2. ® 5. ® 6.

b): 3 ® 1. ® 2. ® 4. ® 5. ® 6.

### 7.2.1.5 Process diagram

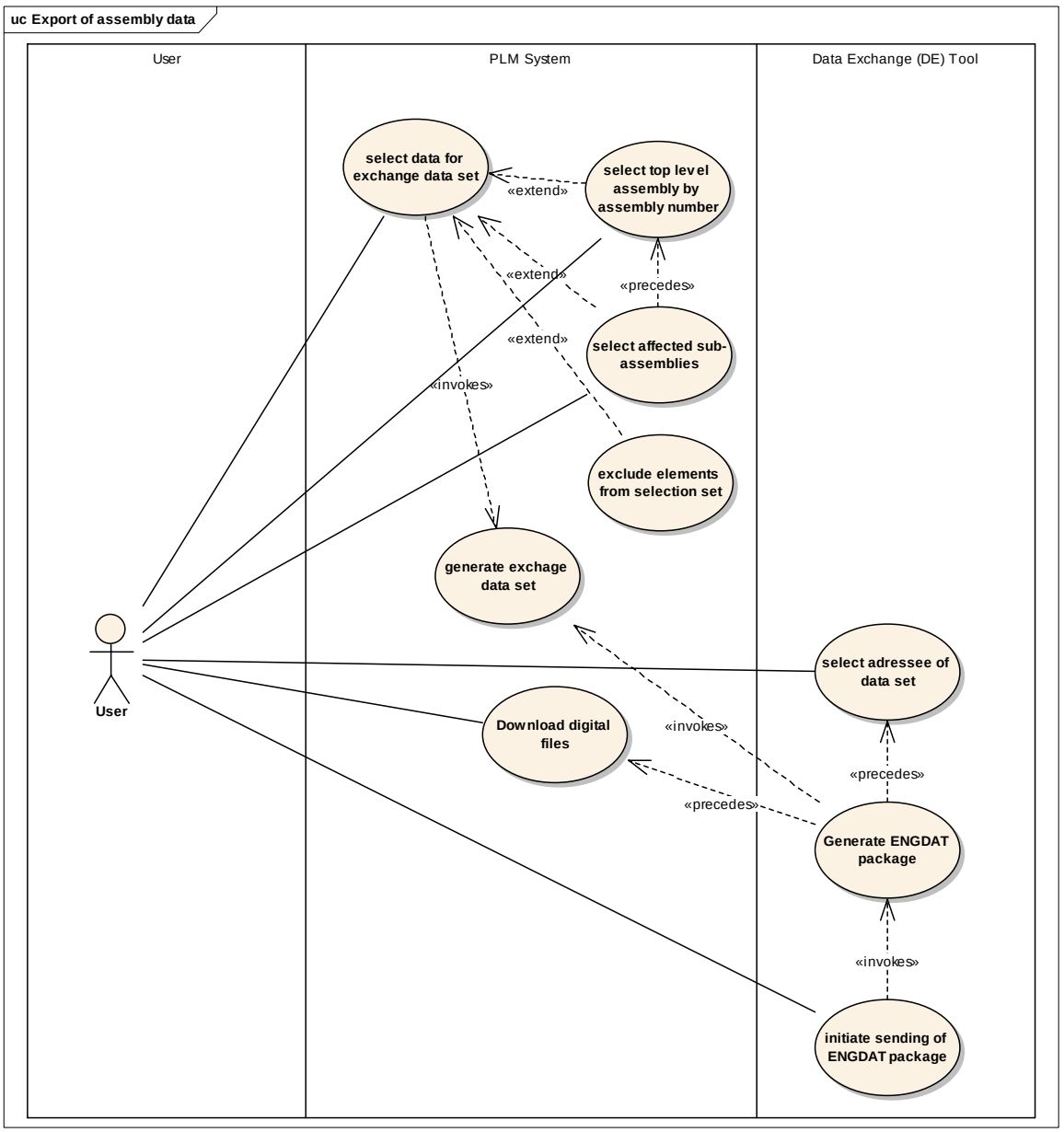


**Figure 7.1 - Use case diagram export of assembly data**

### 7.2.1.6 Process start and end states

Start state / precondition:

The user knows the assembly/part identifiers and digital file (CAD model) identifiers which are supposed to be exported. At least, the identifier of an assembly, which serves as an entry node, is provided. Additionally, a specific “context handle” is known (project, change status, work order, etc.) is known.

Alternative a): Depending on the user environment also a top-level document ID can be the entry node to a structure.

Alternative b): A top-level part and a specific configuration, which controls the way of the expansion of the tree (sub-parts, kind of documents,...), is known.

End state / post condition E1 (Success):

An ENGDAT package including the STEP PDM file and all selected digital files were successfully sent to the addressee.

End state / post condition E2 (Failure):

DE Tool delivers failure notification/report to user. The reasons can be:

- The STEP processor failed.
- The download of files from the PLM system failed.
- The DE Tool failed.

#### 7.2.1.7 Constraints and assertions

Currently the number of STEP files included in one ENGDAT package is recommended to be restricted to one (VDA). Nevertheless, the intention is to allow more than one STEP file per ENGDAT message, see Section 7.2.1.9.

#### 7.2.1.8 Relevant data

- Documents/digital files (CAD files), see Section 8.4
- Document meta data, see Section 8.4
- Assembly/part master data, see Section 8.2
- Assembly structure data (including transformation data), see Section 8.3

#### 7.2.1.9 Topics under discussion / Remarks

Currently no engineering change information is included in the STEP PDM file.

Should more than one STEP file be allowed in an ENGDAT message?

#### 7.2.1.10 Realization with this specification

This specification supports the functionality to realize selection of start nodes and traversing product structures. Especially, the PDTnet queries conformance point provides the necessary methods, see Section 9.8.6 and Section 9.8.7. Retrieving the associated documents is realized by methods as described in Section 9.8.3.

### 7.2.2 Import of assembly data

Import of product data (meta data and geometry) of assemblies and parts from one partner to another via exchange of ENGDAT packages (STEP PDM files, CAD files).

#### 7.2.2.1 Owner of the use case

This use case was defined by Work Group 1 of the PDTnet project.

#### 7.2.2.2 Process purpose

Import of product data which consist of meta data and geometry information of assemblies and its components from one partner to another partner via exchange of ENGDAT formatted packages. The ENGDAT message contains the STEP PDM files and (optionally) the CAD files, in native or neutral format.

#### 7.2.2.3 Partner role descriptions

**Table 7.2 - Roles for import of assembly data**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that processes product data that has been imported. | Person |
| PLM System | Party, that provides the relevant product data and functionality for product data management. This is usually a company's PLM system, which also can be extended by a tool, that provides extended STEP processor functionality. | System |
| Data Exchange (DE) Tool | System, that provides communication with a network and functionality to automatically process and pack/unpack file packages (usually ENGDAT-based). | System |

#### 7.2.2.4 Process definition

The process steps are:

1. The DE tool receives an ENGDAT package.
2. The DE tool unpacks the ENGDAT package and stores STEP PDM and CAD files in defined directories (routing).
3. The PLM system evaluates the received STEP PDM file and displays the included data (assembly data, part data, CAD file meta data) and, optionally, generates an analysis report (comparison of existing data and data to be imported). This step can be initiated by the user or by the DE tool (if it is appropriately integrated). ® see Topics under discussion
4. The user manually processes the data and integrates it into the database of the PLM system or, alternatively, no manual interaction is done. ® see Topics under discussion

The DE tool can notify the user of the import process in different ways, e.g. via e-Mail, via PLM system message, a.s.o.

### 7.2.2.5 Process diagram

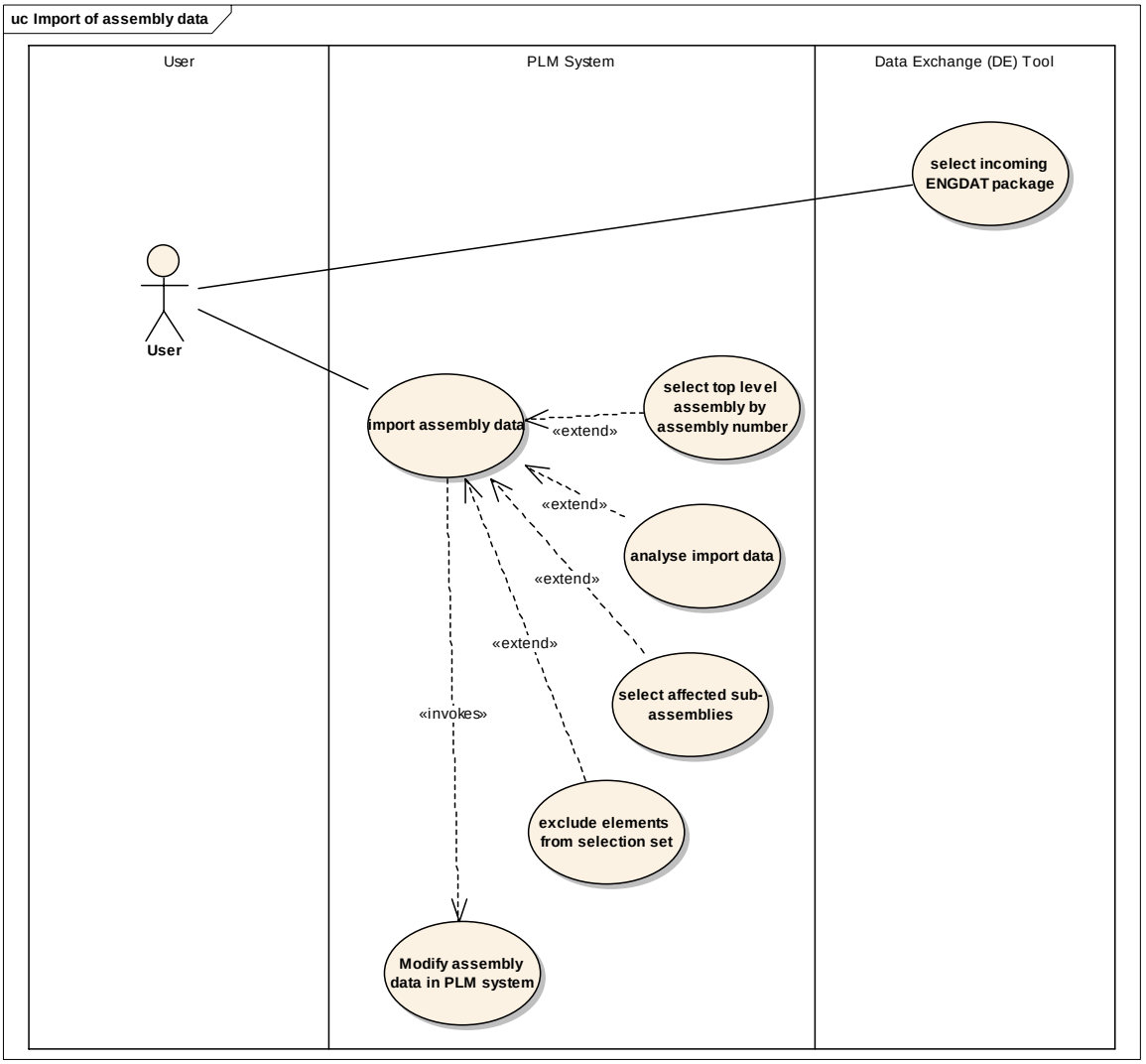


**Figure 7.2 - Use case diagram import of assembly data**

### 7.2.2.6 Process start and end states

Start state / precondition:

- An ENGDAT package including a STEP PDM file and one or more digital files (CAD files,...) has been received successfully. This means:
- The ENGDAT message contains the expected correct data.
- No inconsistencies between STEP file and references to digital files exist. ® see Topics under discussion
- User selected the mode for import (update, create, etc.)

End state / post condition E1 (Success):

The received PDM data has been successfully integrated in the PLM systems' database.

The received CAD files have been successfully stored in the defined storage areas.

Partial incorporation of data in the PLM system, if the user allowed it.

End state / post condition E2 (Failure):

The process results in a failure message. A failure can occur due to the following reasons:

- The ENGDAT message contains errors and can not be processed correctly.
- The STEP PDM file contains errors and can not be processed correctly (syntactically, semantically, e.g. STEP PDM Schema, etc.).
- The loading process into the PLM system caused errors.

#### 7.2.2.7 Constraints and assertions

None.

#### 7.2.2.8 Relevant data

- Documents/digital files (CAD files), see Section 8.4
- Document meta data, see Section 8.4
- Assembly/part master data, see Section 8.2
- Assembly structure data (including transformation data), see Section 8.3

#### 7.2.2.9 Topics under discussion

- Who or which system checks, whether the STEP file and the references to digital files included in an ENGDAT message are consistent? Definition of a separate use case?
- On supplier's side: How to handle product/document meta data, that is not managed by the own PLM system (or no PLM system exists) but that has to be re-exported to the OEM?
- Export of version/status information for re-exported assemblies/parts could be discussed. At the moment no version/ status information is used.
- The CATIA model name must not be changed by a supplier.
- On supplier's side: How to associate product data identified by OEM identifiers to product data in the own PLM system?
- On supplier's side: How to manage different assembly structures?

#### 7.2.2.10 Realization with this specification

This specification supports the functionality to realize selection of start nodes for insertion and traversing and selecting product structures. Especially, the PDTnet queries conformance point provides the necessary methods, see Section 9.8.6 and Section 9.8.7. This functionality is applied both to the assembly data in the exchange package and to the data already stored in the PLM system. Comparison and consolidating of both data sets is not part of this specification. Updating the selected data is performed by the write methods defined in the services as given in Section 9.2.6 or Section 9.2.7.

### 7.2.3 Authentication/Start-Up of session

This process allows a user to be authenticated via a PLM client by one or more PLM server(s).

#### 7.2.3.1 Owner of the use case

This use case was defined by the Work Group 2 of the PDTnet project.

#### 7.2.3.2 Process purpose

This process allows a user to be authenticated via the PLM client by one or more PLM server(s).

#### 7.2.3.3 Partner role descriptions

**Table 7.3 - Roles for authentication and start-up of session**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that wishes to log in a remote PLM server. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server | System |
| PLM server | System, that provides the relevant PLM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.3.4 Process definition

This use case includes the initiation of the connection between PLM client and PLM server, the authentication and personalization of the user. This use case usually initiates all following communication and data transfer between a user, using the PLM client, and a PLM server (also called “site”).

Two alternative authentication processes are possible, which can also be combined:

1. The first attempt to access a remote PLM server will automatically start the authentication process.
2. The user explicitly starts a login procedure to authenticate in one or more PLM server(s) in the beginning of a session.

The following accesses to specific PDM data will be validated within the use case “Authorization.”

#### 7.2.3.5 Process diagram

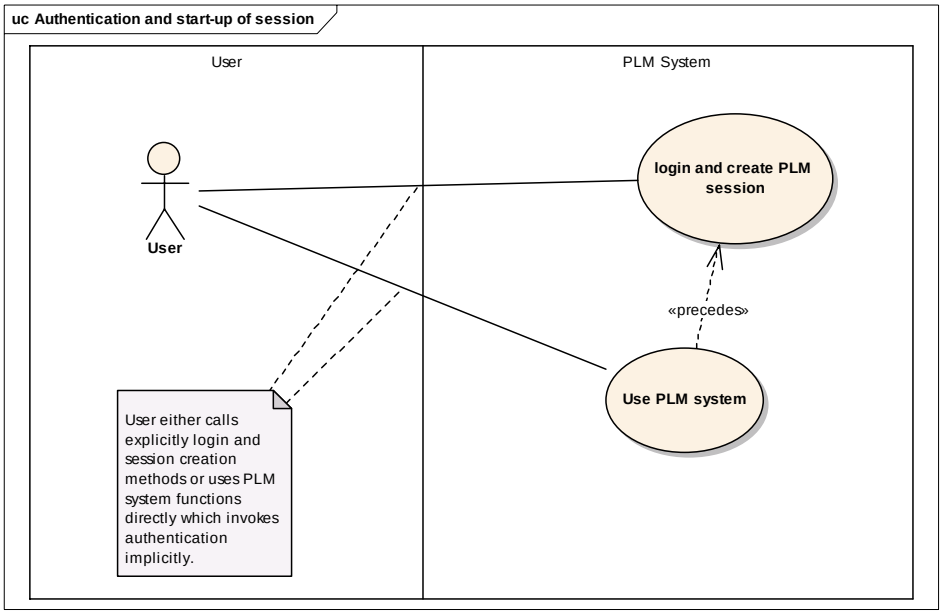


**Figure 7.3 - Use case diagram authentication and start-up of session**

#### 7.2.3.6 Process start and end states

Start state S1:

- The user owns a user name and a password valid for a certain PLM server (site).
- The client provides the necessary site information for the network connection.
- The user knows a valid development project to be authorized to access product data on the PLM server.
- The PLM server provides an authentication service based on user, password and session.

End state E1 (Success):

- The user is successfully logged in and, optionally, the PLM server returns a session id.

End state E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
- The user is not allowed to access the PLM server (return message: “Permission denied”).
- The PLM server itself is not available.

#### 7.2.3.7 Constraints and assertions

A development project defines a project in which persons work together on a certain set of product data. A development project can be a car/vehicle project, a module development project, etc.

#### 7.2.3.8 Relevant data

User name, password, development project, site information (PLM server system), optional: session id.

#### 7.2.3.9 Realization with this specification

This specification recommends two methods to access PLM system functionality

- using PLM_resource_adapter functionality (see Section 9.2.2) to obtain a valid PLM system connection reference via the managed PLM_connection_factory (see Section 9.2.4), or
- directly using an appropriate PLM_connection service instance.

This specification does not define explicit methods to login to a particular system but encourages implementations to use the platform specific security specifications instead - see Section 10.7.4. The handling of additional properties or options to configure access to a PLM systems, like development project, site information and others shall be subject to handling properties as defined in the PLM_service interface, see Section 9.2.1.

### 7.2.4 Authorization

This process validates the access rights of a specific user (designer, group, department, company) to access specific product data on a PLM server.

#### 7.2.4.1 Owner of the use case

This use case was defined by the Work Group 1 of the PDTnet project.

#### 7.2.4.2 Process purpose

This process validates the access rights of a specific user (designer, group, department, company) to access specific product data on a PLM server.

#### 7.2.4.3 Partner role descriptions

**Table 7.4 - Roles for authorization**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that wishes to access PDM data on a remote PLM server. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.4.4 Process definition

This use case describes the authorization process of a user who attempts to request specific product data on a PLM server. It is used by all other use cases (e.g., when extracting product structure trees). The actual process description is dependent on the authorization mechanisms provided by the PLM server.

#### 7.2.4.5 Process diagram

None.

#### 7.2.4.6 Process start and end states

Start state S1:

- A previous authentication process was successful (e.g., by given session id).
- The PLM server provides an authorization service based on user, password and session related to specific product data elements. Additionally, the association of product data elements to a development project has to be supported.
- Specific product data that is requested by a user.

End state E1 (Success):

- The user is identified to have the appropriate rights to access the requested product data. The calling process is enabled to provide the product data to the user.

End state E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reason:
- The user is not allowed to access the requested product data. Since it could be intended to keep the existence of the requested data completely secret, the user should not get the information “Access denied.” Instead, he should get a failure message like “Data not found.”

#### 7.2.4.7 Constraints and assertions

The PLM server provides an authorization service based on user, password and session related to specific product data elements. Additionally, the association of product data elements to a development project has to be supported. The detailed mechanisms of authorizing specific users to access specific product data elements depend on the PLM server's internal authorization features and company specific customizing.

Specific assertions:

- The PLM server manages the association of user/development project to a specific server internal role concept.
- The general role “owner” is provided having all rights for the owned data objects.
- Defined access rights to all other (not owned) data objects are: View, Download, Write, Create.

#### 7.2.4.8 Relevant data

User name, password, development project, optional: session id

- Requested product data, see Section 8.3

#### 7.2.4.9 Realization with this specification

PLM Services does not define its own authorization mechanisms but encourages PSM to use platform specific solutions - see Section 10.7.

### 7.2.5 Start node identification

Identify the start node of a product structure to enable browsing in the product structure.

#### 7.2.5.1 Owner of the use case

This use case was defined by the Work Group 2 of the PDTnet project.

#### 7.2.5.2 Process purpose

Identify the start node of a product structure to enable browsing in the product structure.

#### 7.2.5.3 Partner role descriptions

**Table 7.5 - Roles for start node identification**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.5.4 Process definition

This use case defines the process of identifying the start node of a product structure in a PLM server. The end state / post condition of the use case is the precondition for the start of the use cases “Browsing down/up product structure data.”

The process steps are:

1. User enters ID (part number and optionally part version number) or Wild Card ("*" for "all").
2. PLM client submits search request ® Exception: The PLM server does not respond.
3. PLM server receives ID or Wildcard and triggers search in PLM system ® Exception: The connection between PLM client and PLM server is down.
4. PLM system executes query in its database ® Exceptions: Database is not available, no data found, user is not authorized to access the data, etc.
5. PLM server returns start node and list of views.
6. PLM client displays list of start nodes.

#### 7.2.5.5 Process diagram

None.

#### 7.2.5.6 Process start and end states

Start state / precondition S1:

The user is correctly logged in, connected to the server, positively identified and authorized.

- The service is available.
- The user enters an ID ("Sachnummer" etc.) or wildcard for the structure start node.

End state / post condition E1 (Success):

- List of product structure nodes including their possible views / configurations

End state / post condition E2 (Failure):

- In case of missing authorization: Exception, message: "No items found or access denied."

#### 7.2.5.7 Constraints and assertions

None.

#### 7.2.5.8 Relevant data

- Product structure data, see Section 8.3

#### 7.2.5.9 Topics under discussion

The user should be able to enter either internal or external part master ids ("Alias-Query").

#### 7.2.5.10 Realization with this specification

This functionality is provided by the various query functionality within the Generic Queries Conformance Point, the XPath Conformance Point, the Specific Conformance Point and the PDTnet Conformance Point. See Section 10.6 for examples utilizing the functions in these conformance Points.

### 7.2.6 Browsing down product structure data

This process allows a user starting with the product structure to get a view on all product structure relevant data including document (structure) data that is relevant for this specific user or a specific project, independently of the provider of the data.

#### 7.2.6.1 Owner of the use case

This use case was defined by the Work Group 2 of the PDTnet project.

#### 7.2.6.2 Process purpose

This process allows a user starting with the product structure to get a view on all product structure relevant data including document (structure) data that is relevant for this specific user or a specific project, independently of the provider of the data.

#### 7.2.6.3 Partner role descriptions

**Table 7.6 -Roles for browsing down product structure data**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.6.4 Process definition

This use case includes the browsing of product structure data down a product structure, basic part classification data and associated document meta data. For browsing up a product structure ("where used" query), a separate use case is defined.

The following requirements are defined:

Multiple views on the product structure have to be supported, e.g. lead view, supplier's assembly structure, spare part structure, second tier supplier's view, etc.

- The relationship between different base classification data has to be handled (customer's and supplier's data).
- The assignment of structure and classification data to documents has to be consistent and browsing documents must result always in displaying identical information.
- The user defines a set of parameters (filter information), that specifies characteristics of the desired structure nodes in detail. Filtering the data will be defined as a separate use case “PLM filter.”
- Browsing in different PLM server systems has to be supported. This means, the change of a server site has to be possible (“Multi-site support”) when the user selects a structure node, which links to a supplied item provided by another PLM server. This enables the user to browse into a sub-structure of the development partner (e.g., OEM user browses into sub-structure of supplier or vice versa) and to see the information consistently in one single structure tree. The concept for this mechanism is the following:
  - Reference tables connecting the OEM part identifiers to the supplier part identifiers (“alias identifiers”) are managed by the PLM servers, containing for each exchange node:
    - Own part id (item_version to be supported)
    - Corresponding alias id on PLM server of partner
    - Unique identifier for partner PLM server site: harmonized organization ID (e.g., “bmw.de”).
- An additional reference table for the association of organization id and URL (server site connection) is provided on the PLM client site.

The process steps are:

1. PLM client sends a query for substructure specified by the user to the PLM server
   a. In case of the structure node being a “supplied item,” i.e., the selected structure node represents an alias identifier:
      - Client retrieves alias site connection information (URL) from reference table.

- Client asks user for password for alias site (only in case of first request to this site).
- Client performs Login, Start node query on alias server site using current development project.

Steps repeated by PLM server for each product structure node in the scope of the query:

2. Check authorization regarding requested data ® Exception: Access denied (PLM server).
3. Collect requested data within PLM server.

End of repeated steps.

4. PLM server sends data to PLM client.
5. Display structure and items in PLM client.

#### 7.2.6.5 Process diagram

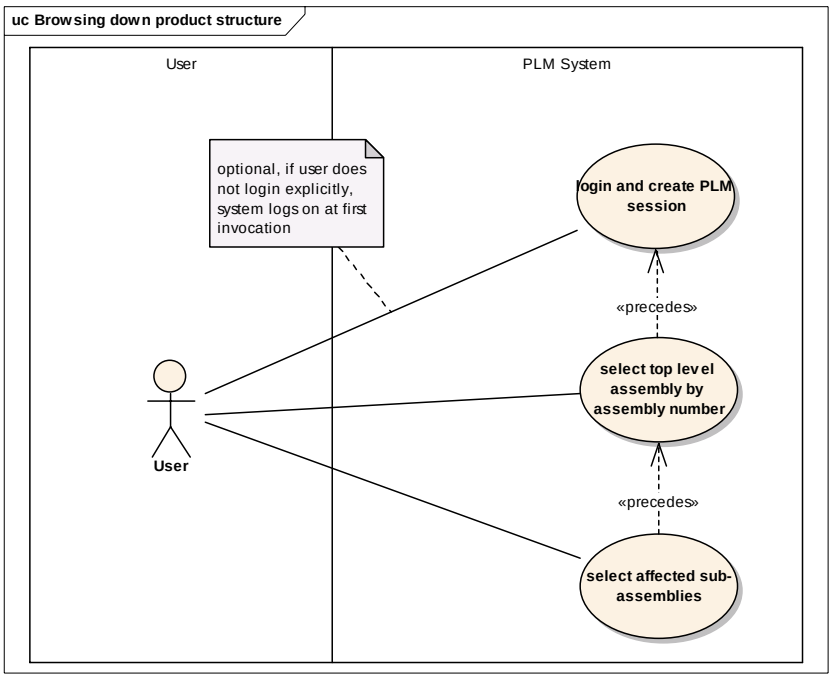


**Figure 7.4 - Use case diagram browsing down product structure data**

#### 7.2.6.6 Process start and end states

Start state / precondition S1:

- A specific development project is defined, which itself defines certain items of product data (e.g., assemblies, parts, documents), that will be subject to change or creation during the project's life time. These items are identified by identifiers.
- The end state / post condition of use case "Start node identification" or one of the children of the start node.
- The user is correctly logged in and authorized to access the requested information.

- The level of depth down the start node / current node is defined (default: 1 level down the current node).
- The necessary filter information is defined, i.e., the result of the use case “PDM filter” is provided.

End state / post condition E1 (Success):

- The process results in a filtered list or a structure tree containing at least the identifiers of product data items, and additional information about the items (e.g., URLs to documents or additional item information to be downloaded).

End state / post condition E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
- The user is not authorized to access the data.
- The requested data is not available on the PLM server.

#### 7.2.6.7 Constraints and assertions

If process step 2 leads to an exception regarding a specific structure node, the whole process must continue. The structure node affected by the exception is not included in the collected data set.

#### 7.2.6.8 Relevant data

Product structure data

- Basic part classification data, see Section 8.2
- Document meta data, see Section 8.4

#### 7.2.6.9 Realization with this specification

This specification supports the query for a particular start node, part or document. This query may contain filters for specific attribute values expressed by location_path, attribute and predicates as described in Section 9.4. The exception handling is described in Section 9.2.10.

### 7.2.7 Browsing up product structure data

This process allows a user starting with the product structure to get a view on all product structure relevant data including document (structure) data that is relevant for this specific user or a specific project, independently of the provider of the data.

#### 7.2.7.1 Owner of the use case

This use case was defined by the Work Group 1 of the PDTnet project.

#### 7.2.7.2 Process purpose

This process allows a user starting with a specific product structure node to get a view on all relevant product structure nodes in which this specific node is included (“Where used” query). For browsing down a product structure, a separate use case is defined.

#### 7.2.7.3 Partner role descriptions

**Table 7.7 - Roles for browsing up product structure data**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.7.4 Process definition

This use case includes the browsing of product structure data up a product structure (“where used” query).

The following requirements are defined.

Multiple views on the product structure have to be supported, e.g. lead view, supplier's assembly structure, spare part structure, second tier supplier's view, etc.

- The user defines a set of parameters (filter information), that specifies characteristics of the desired structure nodes in detail.

The process steps are:

1. PLM client sends a query for “where used” nodes specified by the user to the PLM server

Steps repeated by PLM server for each product structure node in the scope of the query:

2. Check authorization regarding requested data ® Exception: Access denied (PLM server)
3. Collect requested data within PLM server

End of repeated steps.

4. PLM server sends data to PLM client
5. Display structure and items in PLM client. The way of presentation and needed interaction have to be defined by the application projects.

### 7.2.7.5 Process diagram

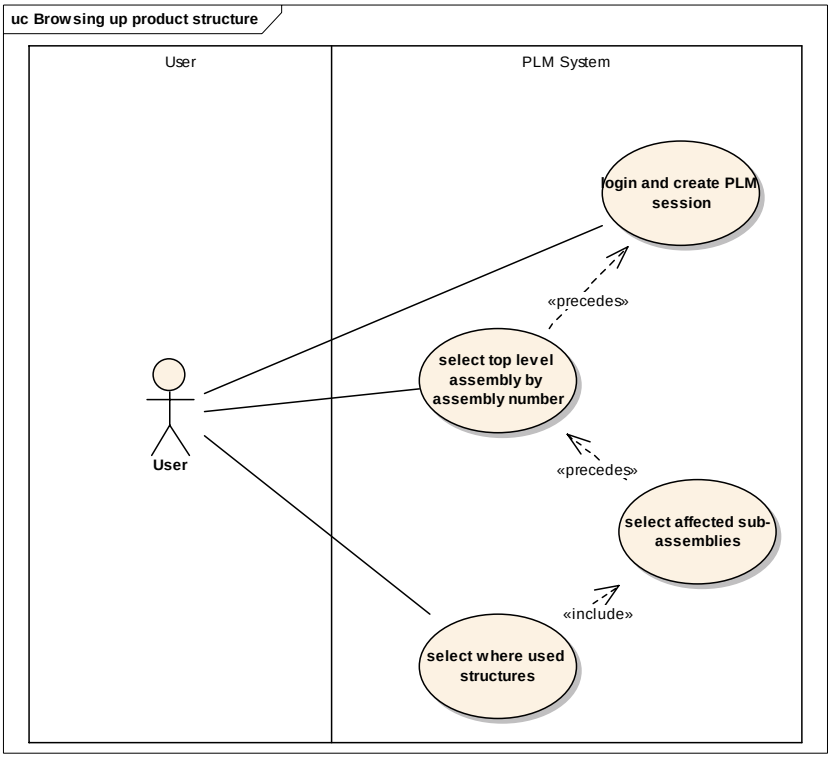


**Figure 7.5 - Use case diagram browsing up product structure**

### 7.2.7.6 Process start and end states

Start state / precondition S1:

A specific engineering development project is defined, which itself defines certain items of product data (e.g., assemblies, parts, documents) that will be subject to change or creation during the project's life time. These items are identified by identifiers.

- The end state / post condition of use case “Start node identification” or one of the children of the start node, that means item or item_version. Item_version is optionally in order to enable the access of versioning information starting from the part number. Additionally, the single_instance can be identified (maybe by user interaction). This is “nice to have” in general, but required as precondition for a “Search in design space” functionality.
- The user is correctly logged in and authorized to access the requested information.
- The level of depth up the start node / current node is defined and restricted to direct parent or root node (default: direct parent node).

The necessary filter information is defined, i.e., the result of the use case “PLM filter” is provided.

End state / post condition E1 (Success):

- The process results in a filtered list or a structure tree containing only identifiers of product data items (root nodes or direct parent nodes). Only structure nodes which the user is authorized to see are included.

End state / post condition E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
  - The user is not authorized to access the data.
  - The requested data is not available on the PLM server.

#### 7.2.7.7 Constraints and assertions

Whenever the PLM System is providing a single_instance concept, the start node used may be the single_instance. If the single_instance is used, there is no necessity for repeating process steps 2 and 3. This statement needs to be evaluated!

- The level of depth up the start node / current node is defined and restricted to direct parent or root node (default: direct parent node).
- Exactly one root node exists for one development project.
- Need of unique filter, that displays the root node only once.
- Within the Client GUI the change to one of the resulting development project (in case of a result list containing root nodes) should be possible.

#### 7.2.7.8 Relevant data

- Product structure data, see Section 8.3.

#### 7.2.7.9 Realization with this specification

This specification supports the query for a particular part start node, see Section 9.8.6. This query may contain filters for specific attribute values expressed by location_path, attribute, and predicates as described in Section 9.4. The exception handling is described in Section 9.2.10.

### 7.2.8 Download of product data

#### 7.2.8.1 Owner of the use case

This use case was defined by the Work Group 1 of the PDTnet project.

#### 7.2.8.2 Process purpose

This use case has to be described under consideration of two main criteria:

What product data is to be downloaded?

- Download of a single digital file: either geometry (CATIA, STEP) or other binary formats (e.g., TIFF)
- Download of a set of digital files
- Download of structures including optionally digital files
- Download of product meta data of a (structure) node

How is the product data to be downloaded?

- Using online download: via HTTP, only for available documents - no conversion functionality provided
- Using offline download (e.g., via OFTP)

Due to these distinctions the use case “Download of product data” is divided into two use cases, which are described in Section 7.2.9 and Section 7.2.10.

### 7.2.9 Download meta data including structures

This use case allows the user to identify meta data including structures that he wants to store in a local file system, or that he wants to import into an own PLM system. The format of the transferred data differs:

Online download: The data is transmitted as an data stream (e.g. SOAP message response for web services based implementation). File representations are not supported in this case.

Offline download: The data is sent as an file within the download package. It can be a STEP AP214 Part21, which is specified in the server configuration and considers requirements at target side.

If the detail level covers digital documents, the download of these files will be initiated. The download of existing Part 21 files is not covered by this use case either. For this, see use case “Download of a single digital file.” If the data is sent offline, the files may be added to the download package, which is specified in the server configuration and considers requirements at target side.

This functionality covers the access of multiple PLM server Interfaces. For this, two possibilities exist:

1. The user has access to the PDM data of his direct (!) partners. This is covered by the use cases.
2. All other alternate possibilities are managed by the PLM server interface (e.g., data in a 2nd-tier supplier’s PLM system).

#### 7.2.9.1 Partner role descriptions

**Table 7.8 - Roles for downloading meta data including structures**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.9.2 Non-functional requirements

The following requirements with respect to the design of the PLM client GUI are defined:

- The level of detail (“configuration”) can be defined depending on the application project. The technology for defining this configuration is not defined yet.

- The approval status of the relevant data has to be managed by the PLM server interface via authentication and authorization use cases.
- The user is not able to exclude single objects that belong to the tree defined by the start node.
- An additional use case is needed: “PLM Filter”. This use case enables the user to define some special properties that restrict the following amount of managed data.

#### 7.2.9.3 Process definition

The standard process consists of the following steps (the steps directly refer to elements of the user interface of the PLM client):

1. Using the context menu (“right mouse click”) for starting the use case. The user may use this menu only for items and documents in order to be STEP compliant in any cases.
2. By identifying the menu button “download of meta data” a sub-menu appears that provides all available levels of detail (called “configurations”): download of part master data, download of part and document master data, etc.
3. The user identifies the wished level of detail using the sub-menu.
4. If the user defined to download structure information the next sub-menu appears: “Level of structure depth.”
5. In the right frame a list of items appears that were defined for the download process. The user is able to use a scroll bar for browsing through the list.

Optionally: If the download information was not already received by the client, the following steps will be performed:

6. The client is calling the PLM server using a specified query.
7. The server generates the product data and sends the resulting data stream to the client interface.

Mandatory:

8. The User starts the download by choosing the Online or Offline Download entry in the right click menu.

Online Download:

9. The PLM client sends a query to the PLM server.
10. The PLM server sends the requested data as a data stream to the PLM client.
11. The client takes the data stream and:
12. calls the “Upload Query” to the second PLM system or
13. writes an data file.

Offline Download (see also “Initiation of an Offline Download”):

14. The PLM client sends a query to PLM server interface or to the involved EDI-Tool ® Input to use case “Initiation of an Offline Download.”
15. A Client notification is created by the EDI-Tool.

#### 7.2.9.4 Process diagram

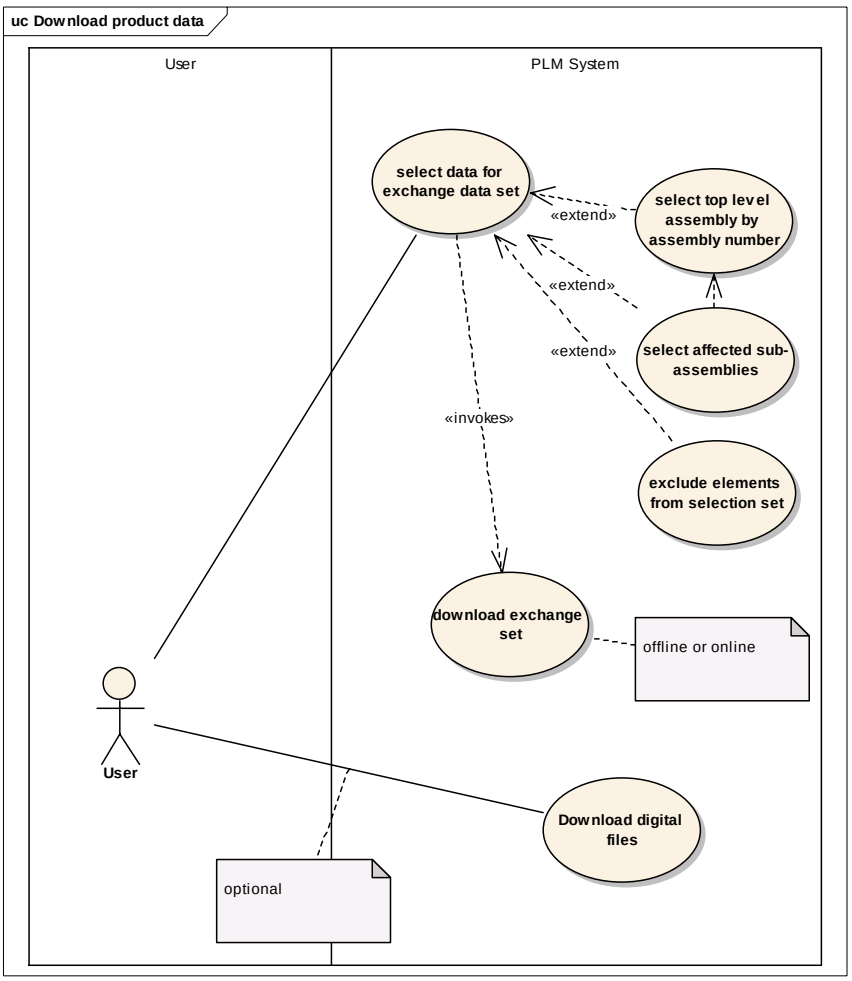


**Figure 7.6 - Use case diagram download of product data**

#### 7.2.9.5 Process start and end states

Start state S1:

- Successful results of Authorization and Browsing use cases.

End state E1 (Success):

- Offline Download: A notification of an additional exchange process is provided (e.g., “Off-line transfer is running”).
- Online Download: A notification for the User, if the download is finished (with success or not).
- The selected meta data including structures is stored in a data file on a local computer (file system), or generated as data stream as input for the Upload use case.

End state E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
  - The user is not authorized to access the PLM server.
  - The PLM server interface detected a problem.
  - The user is not authorized to download the requested data.
  - The PLM server itself is not available.
  - Offline Download: Triggering the EDI-Tool failed.
  - Online Download: Not sufficient disc space for storing the file.

#### 7.2.9.6 Relevant data

- All product data (part master, document master, etc.), see Section 8.2 and Section 8.4

#### 7.2.9.7 Topics under discussion / Remarks

- This download use case ends by creating a data file or a data stream. This data can be re-used by Upload Use Cases.
- Definition of “configurations”: Should they be based on transformation rules?

#### 7.2.9.8 Realization with this specification

See Section 7.2.1.

### 7.2.10 Download a single digital file

This process allows a user to download a single specific digital file (geometry file, TIFF, etc.) from a remote PLM server to a local storage. The download also includes the viewing of digital files, as far as a viewing tool is automatically started on the user side after the download process has finished. This process is called “simple viewing.”

#### 7.2.10.1 Partner role descriptions

**Table 7.9 - Roles for downloading a single digital file**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.10.2 Process definition

This use case includes the identification of a single digital file to be downloaded, the start, the monitoring of the progress, and the check of the success of the data transport from a PLM server to a local storage.

The process steps are as follows:

1. The user identifies the digital file to be downloaded from the PLM server.
2. The User starts the download by choosing the Online or Offline Download entry in the right click menu.

Online Download:

3. The PLM client sends a query to the PLM server.
4. The PLM server sends the requested digital file data to the PLM client.
5. The PLM client receives the digital file and displays it directly, opens an external application to display it or let the user store it in the local file system.
6. A notification is sent to the User (in case of success and in case of failure).

Offline Download (see also “Initiation of an Offline Download,” see Section 7.2.2):

7. The PLM client sends a query to PLM server interface or to the involved EDI-Tool ® Input to use case “Initiation of an Offline Download.”
8. For the file export from the PDM Vault a copy of the document should be created, no file locking mechanism (for parallel use by other users) should be implemented. The export could be triggered by the PLM server or by the EDI-Tool.
9. A Client notification is created by the EDI-Tool.

#### 7.2.10.3 Process diagram

None

#### 7.2.10.4 Process start and end states

Start state S1:

- The user has been successfully authenticated.
- The user is authorized to know that the digital file exists.
- The user has got a list or a structure tree containing at least the identifier of the digital file and an appropriate URL.
- The kind of the access (viewing, changing) is specified. Currently only viewing functionality is considered.
- The final trigger is the selection in the context sensitive menu (“Download selected file online/offline”) that belongs to a selected single digital file.

End state E1 (Success):

- Offline Download: A notification of an additional exchange process is provided (e.g., “Offline transfer is running”).
- Online Download: A notification for the User, if the download is finished (with success or not).
- The digital file, that has been specified by the user for download, is opened and displayed or stored on the local storage.

End state E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:

- The user is not authorized to access the PLM server.
- The user is not authorized to download the digital file.
- The requested digital file is not available on the PLM server.
- The PLM server itself is not available.
- Offline Download: Triggering the EDI-Tool failed.
- Export (checkout) functionality failed (digital file doesn't exist, the file is already used by an other user).
- Online Download: Not sufficient disc space for storing the files.

#### 7.2.10.5 Constraints and assertions

- The downloaded file is always not compressed if it is sent online. Then the file can be opened directly and may be viewed using a client plug in or an external application. Compression is only allowed if an offline transfer process implies a package mechanism
- The file name is generated by server/system specific rules.

#### 7.2.10.6 Relevant data

- Document meta data, see Section 8.4
- Document data (digital file), see Section 8.4

#### 7.2.10.7 Realization with this specification

Download functionality is assumed to be implemented in the PSM. A basic methodology to access the necessary URL of the desired objects is provided by the methods of the basic interface for the PLM_connection realizations, see Section 9.2.5.

### 7.2.11 Generic object query

This use case allows a user to generically access objects (e.g., items, documents) as result of a specified filter condition. Feasible filter parameters and the functionality for the collection and provision of these objects have to be provided by the PLM server. Therefore, this generic use case can be specialized to further detailed use cases. Examples for detailed use cases are:

- Find all parts contained in a design space by providing bounding box parameters.
- Find heat sensitive parts by providing temperature parameters.

#### 7.2.11.1 Owner of the use case

This use case was defined by the Work Group 2 of the PDTnet project.

#### 7.2.11.2 Process purpose

This use case allows a user to generically access objects (items, documents) as result of a specified filter condition. Feasible filter parameters and the functionality for the collection and provision of these objects have to be provided by the PLM server. Therefore, this generic use case can be specialized to further detailed use cases. Examples for detailed use cases are:

- Find all parts contained in a design space by providing bounding box parameters.
- Find heat sensitive parts by providing temperature parameters.

#### 7.2.11.3 Partner role descriptions

**Table 7.10 - Roles for generic object query**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that wishes to request information. This could be a person, who interacts with the PLM client, or a system that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.11.4 Process definition

The process steps are:

1. User chooses the intended (and provided) functionality (specialized query).
2. User defines a development project or uses the existing one.
3. PLM client displays the parameter names, that have to be provided to filter out the correct data within the PLM server, according to the chosen functionality (see step 1).
4. User provides required parameter values (objects properties, bounding box information, etc.) and initiates query to PLM server interface (single PLM Interface).
5. PLM System is processing the query that results in an object list.
6. Object list is displayed within the PLM client.

### 7.2.11.5 Process diagram

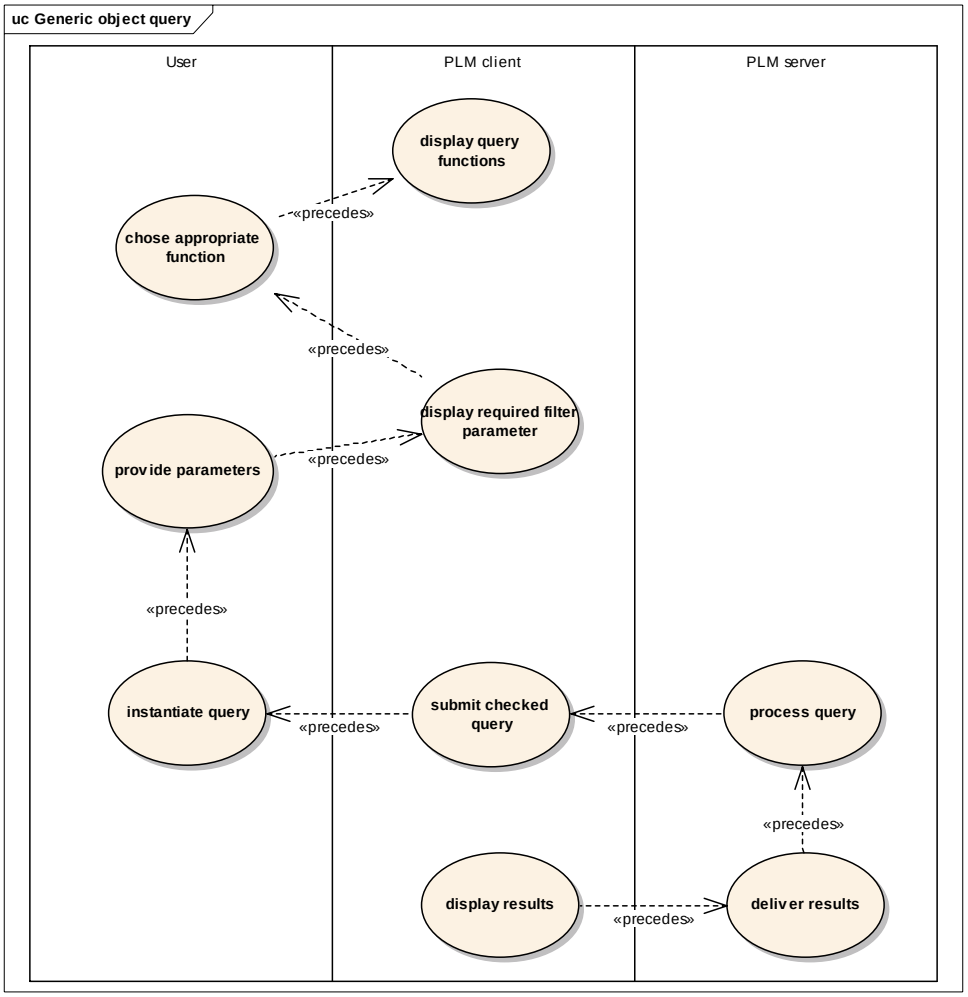


**Figure 7.7 - Use case diagram for generic object query**

### 7.2.11.6 Process start and end states

Start state S1:

- The authentication and authorization of the user was successful.
- A valid development project is existing.
- The available specialized types of object queries related to specific objects have been previously submitted by the PLM server (see use case “Start-up of session”).

End state E1 (Success):

- List of objects that were requested according to the specialized query and filter parameters. Example for specialized query “Search in design space”: All parts contained in the defined design space as a list of items.

End state E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
  - No development project defined.
  - The user is not authorized to access the data (see also use case “Authorization”).
  - The requested data is not available on the PLM server.
  - ·Functionality is not supported for this object type.

#### 7.2.11.7 Constraints and assertions

Only one single PLM server is accessed. A generic object query that is sent simultaneously to more than one PLM server is not supported.

#### 7.2.11.8 Relevant data

- Product structure data, see Section 8.3
- Basic part classification data, see Section 8.2
- Document meta data, see Section 8.4
- Document data, see Section 8.4

#### 7.2.11.9 Realization with this specification

The generic object query functionality is defined in the Generic Queries Conformance Point, see Section 9.4.

### 7.2.12 Search in design space

This use case is a specialization of the use case “Generic object query,” see Section 7.2.11.

#### 7.2.12.1 Process purpose

Purpose of the “Search in design space” process is to query all parts which are located in the neighborhood of a given part. This use case allows a designer at the supplier site to search for parts which are positioned in a certain area around a specified part. The calculation of the neighborhood relation of parts will be done by using the “bounding boxes” of the parts. The user should be able to “blow up” the bounding box around a part in order to get all parts in a certain distance of the given part.

#### 7.2.12.2 Partner/actor role descriptions

**Table 7.11 - Roles for search in design space**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person who interacts with the PLM client, or a system that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.12.3 Process definition

The process could be seen as a query in which the query parameters do not exist as discrete PDM data in the PLM system. Actually, the criteria for the evaluation of the result set is the geometrical relation between the given part and all other parts in a given assembly. For example, the designer has to modify the design of the oil pump of a car. He needs to know which parts are located near to the pump to be able to check whether the modified pump fits into the space left for this device. With the search described here, he can find those parts easily. This use case would probably only be relevant for the OEM side of the PDTnet project.

The following requirements are defined:

- The parts found during the search are displayed in form of a "virtual container" which contains all parts meeting the design space criteria. The virtual container is an assembly which is only created temporarily and which does not represent any form of a real assembly. It is only meant as a set of objects and therefore can be displayed as an assembly with one and only one level.
- It should be possible to combine different search criteria (search in design space, search by defining PDM data filters). For example, all temperature sensitive parts in a certain distance of a hot part have to be found by the query.
- In order to ensure the clearness of visualization, the formerly displayed structures should be made available by means of a "Pull down list" or by "Tabs" which allow to go directly to the assigned structure display.
- The resulting set of items should allow to perform a download (online or offline) on certain items selectable by the user.
- The user should optionally be able to define an assembly ("Start node") in which the parts to find are contained. For example, all parts in a combustion engine should be found.
- Another option is to enter the depth of search, the levels of deepness in an assembly.

### 7.2.12.4 Process diagram

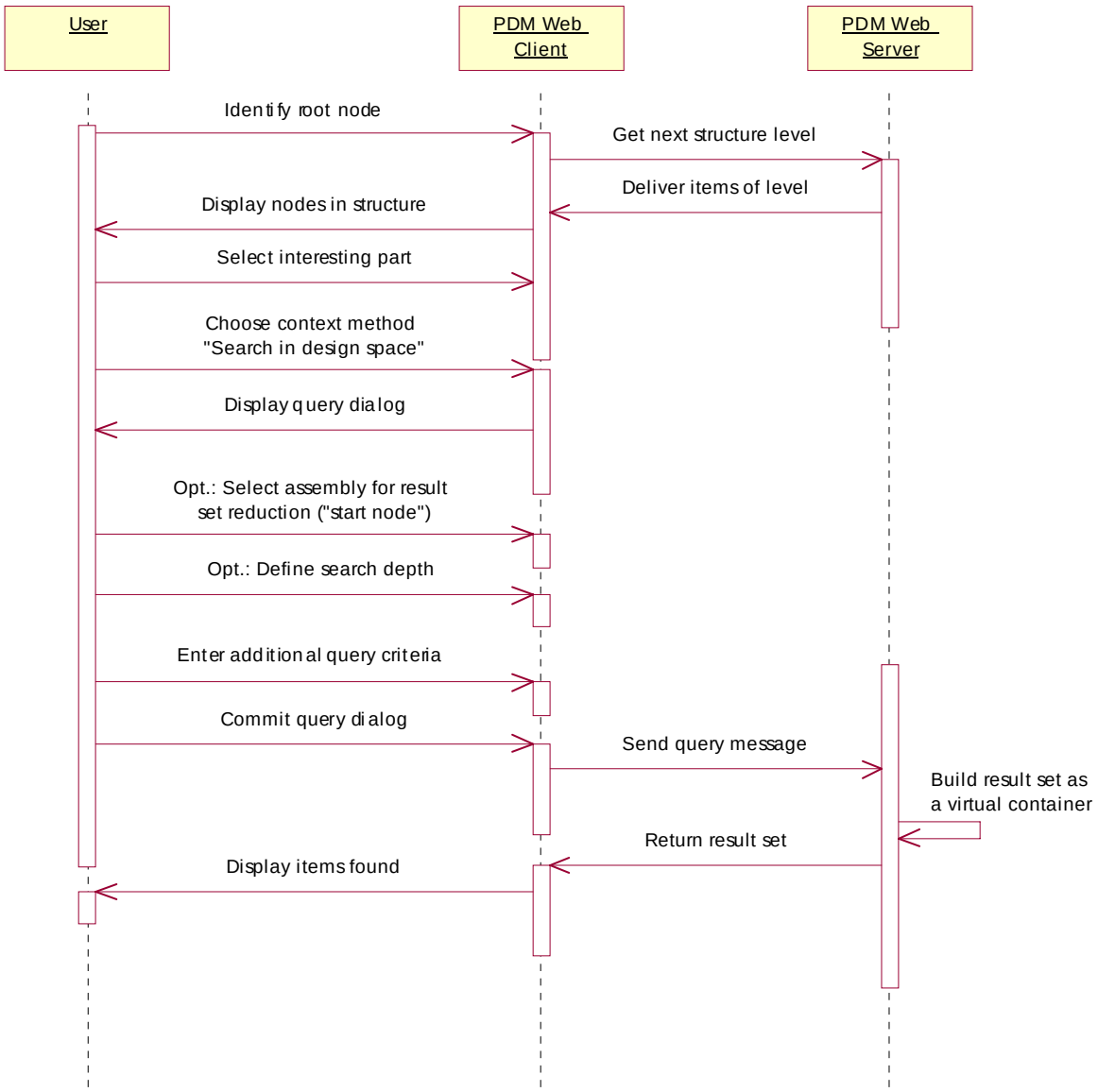


**Figure 7.8 - Sequence diagram for search in design space**

### 7.2.12.5 Process start and end states

Start state / precondition S1:

A specific engineering project is defined, which itself defines certain items of product data (e.g., assemblies, parts, documents) that will be subject to change or creation during the project's life time. These items are identified by identifiers.

- The end state / post condition of use case “Start node identification” or one of the children of the start node, that means an item.
- The user is correctly logged in and authorized to access the requested information.

The necessary filter information is defined (Use case “Generic object query”, see Section 7.2.11).

End state / post condition E1 (Success):

- The process results in a virtual container (see Section 7.2.12.3) containing all the accessible parts found during the query. The number of parts found is displayed.
- The virtual container contains the transformation matrices of the parts in relation to the car origin.
- If no parts or accessible parts were found, an empty virtual container is presented. The number of parts found is displayed, in this case it is 0.

End state / post condition E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
- The selected part contains no geometry. Therefore, there is no possibility to find any parts in the neighborhood of the part. This should be reported by the message “Part contains no geometry.”

#### 7.2.12.6 Constraints and assertions

- The selected part has to contain any geometry as a base for the query.

#### 7.2.12.7 Relevant data

- Product structure data, see Section 8.3

### 7.2.13 Upload of product data

#### 7.2.13.1 Process purpose

This use case allows a user to upload specific product data that was created or changed on a local storage to a remote PLM server.

This use case corresponds mainly to use case “Download of product data,” see Section 7.2.8. Additionally, it requires two functions:

- Identification of correct structure nodes for the integration of uploaded data.
- Creation/change of structures and/or structure nodes, if appropriate.

This functionality is closely related to the underlying access authorization concept. Due to the variety of PLM system specific access authorization architectures this topic is closely depending on the PLM system functionality and/or company specific PLM system usage restrictions.

#### 7.2.13.2 Owner of the use case

This use case was defined by the Work Group 2 of the PDTnet project.

#### 7.2.13.3 Realization with this specification

The requested functionality is provided by the Generic Queries Conformance Point with the write method of the PLM_general_connection or PLM_message_connection, see Section 9.2.6 and Section 9.2.7. Specialized functions are defined in the Specific Queries and the PDTnet Queries Conformance point, see Section 9.7 and Section 9.8, respectively.

### 7.2.14 Upload a single digital file (simple user interaction)

#### 7.2.14.1 Process purpose

This process allows a user to upload a single file which was created or changed on a local storage to a remote PLM server.

#### 7.2.14.2 Process definition

This use case corresponds mainly to use case “Download of a single digital file” (see Section 7.2.10). Additionally, it requires two functions:

Identification of the correct structure node for the integration of uploaded data.

- Creation/change of structures and/or structure nodes, if appropriate. This functionality is closely related to the underlying access authorization concept. Due to the variety of PLM system specific access authorization architectures this topic is closely depending on the PLM system functionality.

#### 7.2.14.3 Process diagram

None

#### 7.2.14.4 Partner role descriptions

**Table 7.12 -Roles for uploading of a single digital file (simple user interaction)**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that wishes to store PDM data on a remote PLM server. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. The PLM system can be extended by a Web Server to build the complete PLM server. | System |

#### 7.2.14.5 Process start and end states

Start state S1:

- The user has got a single file stored on his local file system to be uploaded.
- The user knows the correct structure node in the database of the PLM server for the integration of the data.

End state E1 (Success):

- Offline Upload: A notification of an additional exchange process is provided (e.g., “Offline transfer is running”).
- Online Upload: A notification for the User, if the upload is finished (with success or not). The displayed target structure is refreshed on the screen.
- The file, that had been specified by the user for upload, is stored on the remote PLM server and attached to the target structure. Maybe some new structure nodes were created to attach the file to.

End state E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
  - The user is not authorized to access the PLM server.
  - The user is not authorized to upload the digital file.
  - The user is not authorized to create needed structure nodes.
  - The server can't create needed structure nodes with default values.
  - The specified data could not be integrated in the database of the PLM server (e.g., the correct structure node for data integration could not be identified).
  - The PLM server itself is not available.
  - Offline Upload: Triggering the EDI-Tool failed.

#### 7.2.14.6 Constraints and assertions

The uploaded file is always not compressed. Compression is only allowed if an offline transfer process implies a packaging mechanism.

The target element to assign an uploaded file to can be of type “Item_version” or “Document_version.” In case of a “Document_version” the file can be assigned directly. If an “Item_version” is selected, the server has to create a document with default values to assign the file to. If any creation is not possible, the action fails and the user is notified.

Any directives/parameters for the upload process are stored at server side.

#### 7.2.14.7 Relevant data

Product structure data

- Document meta data
- Document data (digital file)

### 7.2.15 Upload meta data including structures

#### 7.2.15.1 Process purpose

This process allows a user to upload meta data including structures to a remote PLM server. This data was created or changed on a local storage or is the result of a download process.

#### 7.2.15.2 Process definition

This use case corresponds mainly to use case “Download of meta data including structures” (see Section 7.2.9). Additionally, it requires two functions:

- Identification of correct structure nodes for the integration of uploaded data.
- Creation/change of structures and/or structure nodes, if appropriate. This functionality is closely related to the underlying access authorization concept. Due to the variety of PLM system specific access authorization architectures this topic is closely depending on the PLM system functionality.

#### 7.2.15.3 Process diagram

None

#### 7.2.15.4 Partner role descriptions

**Table 7.13 -Roles for uploading meta data including structures**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that wishes to store PDM data on a remote PLM server. This could be a person who interacts with the PLM client, or a system that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.15.5 Process start and end states

Start state S1:

- The user has got data stored on his local file system or stored temporarily as a result of a download process.
- The user knows the correct structure nodes in the database of the PLM server for the integration of the data.

End state E1 (Success):

- Offline Upload: A notification of an additional exchange process is provided (e.g. “Offline transfer is running”).
- Online Upload: A notification for the User, if the upload is finished (with success or not). The displayed target structure is refreshed on the screen.
- The data, that had been specified by the user for upload, is stored on the remote PLM server and integrated into the target structure.

End state E2 (Failure):

- The process results in a failure message. A failure can occur due to the following reasons:
  - The user is not authorized to access the PLM server.
  - The user is not authorized to upload the data.

- The specified data could not be integrated in the database of the PLM server (e.g. the correct structure nodes for data integration could not be identified).
- The PLM server itself is not available.
- Offline Upload: Triggering the EDI-Tool failed.

#### 7.2.15.6 Constraints and assertions

The new structure is sent as message set to the server. The data can be assigned to one or more target elements. If the whole uploaded structure should be assigned to one single element, this will be selected within a message parameter. If there are more complex relations between the new and target elements, the message set also contains the target elements and the relationships to them. In case of an offline transfer, the message set can be replaced by a STEP Part 21 file, which is specified in the server configuration and considers requirements at target side. In case of an online transfer, STEP Part 21 is not supported.

Referenced files has to be uploaded separately using the use cases “Upload a single digital file” or “Upload a set of digital files.” If the data is sent offline, the files may be added to the upload package, which is specified in the server configuration and considers requirements at target side.

Any directives/parameters for the upload process are stored at server side.

#### 7.2.15.7 Relevant data

Product structure data

- Basic part classification data
- Document meta data
- Document data

### 7.2.16 Change notification

#### 7.2.16.1 Process purpose

The designer of a part needs notification when a change to a part happens which affects one of the parts he is responsible for. This could take place when a part in the neighborhood of a given part is changed in its dimensions or properties or when a part in an assembly is moved to another place than before. The user specifies the parts on which he wants to be notified by using the functionality of subscribing specified in use case “Change content of subscription list.”

### 7.2.16.2 Partner/actor role descriptions

**Table 7.14 -Roles for change notification**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person who interacts with the PLM client, or a system that triggers the PLM client. | Person / System |
| E-Mail Client | System, that is able to maintain the user's e-mail. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

### 7.2.16.3 Process definition

Target of the process is the evaluation of objects being changed since the last visit of the user to this object. When a modification of those object is being detected, an appropriate message has to be delivered to the user. Objects could be parts (and part versions), documents (and document versions), or models.

Changes to report could be:

- Creation of a new version of an object
- Change of the release status of an object
- Objects are deleted
- Geometry has changed
- Properties have changed

The following requirements are defined:

- Two possibilities of detecting changes on the server side are conceivable. Which of them is used is depending on the PLM server implementation:
- Whenever an object linked to anybody's subscription list is changed, an e-mail is sent to the user(s).
- In certain periods of time, the subscription lists of all users are checked against the objects they include. When a modification of a certain object is detected, an e-mail is sent to the user.
- The frequency and content of e-mail notifications (confidential data must not be included!) are defined server-specifically.

#### 7.2.16.4 Process diagram

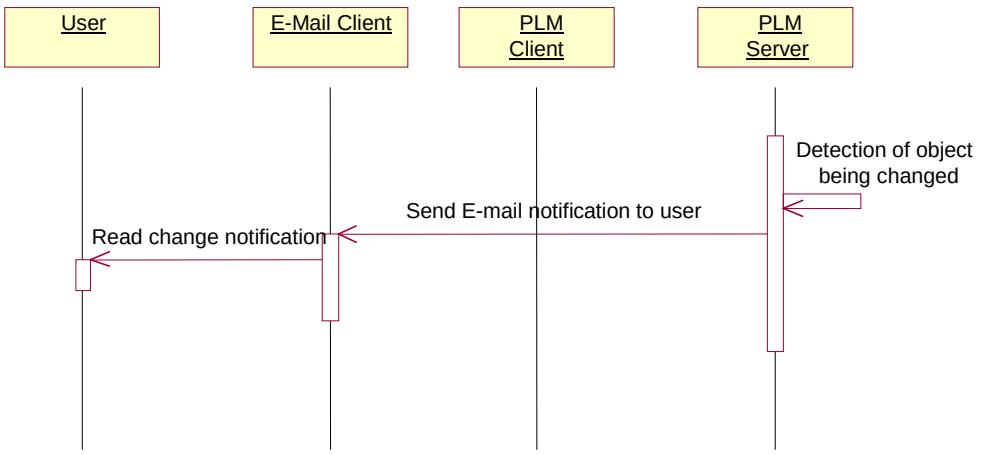


**Figure 7.9 - Process diagram for change notification**

#### 7.2.16.5 Process start and end states

Start states / preconditions S1 and S2:

- User has access to his e-mail client.

End state / post condition E1 and E2 (Success):

- An e-mail notification about changes to one of his objects collected in the clipboard is sent to the user.

#### 7.2.16.6 Constraints and assertions

Currently none are defined.

#### 7.2.16.7 Relevant data

- Product meta data

### 7.2.17 Display content of subscription list and confirm changes

#### 7.2.17.1 Process purpose

To get an overview about objects being changed on the PLM server, the user should be able to display the contents of his subscription list in which he collects all the objects to track. The changed objects should be displayed in an emphasized style to show the status of being changed.

The current content of the subscription list including notifications of changes can be requested by the PLM client:

- when logging in at the server
- when interactively initiated by the PLM client user.

#### 7.2.17.2 Partner/actor role descriptions

**Table 7.15 - Roles for displaying content of subscription list and confirm changes**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.17.3 Process definition

Target of the process is the evaluation of objects being changed on the PLM server since the last visit of the user and the notification of the user by displaying the content of the subscription list. When a modification of those objects is being detected, the objects are marked as changed in the subscription list and the reasons of the changes are displayed.

The following requirements are defined:

- The user controls the start of the evaluation process via the client. The results of the evaluation process are displayed directly in the client.
- The change notification data is transferred by the PLM server using the data constructs provided by AP214 (work management information). An additional transfer of change management/notification documents (like PDF files) is currently not needed.
- The user must be able to define and to modify the content of his subscription list (see use case “Change content of subscription list”).
- The subscription list should be represented as a separate folder within the PLM client GUI.

### 7.2.17.4 Process diagram

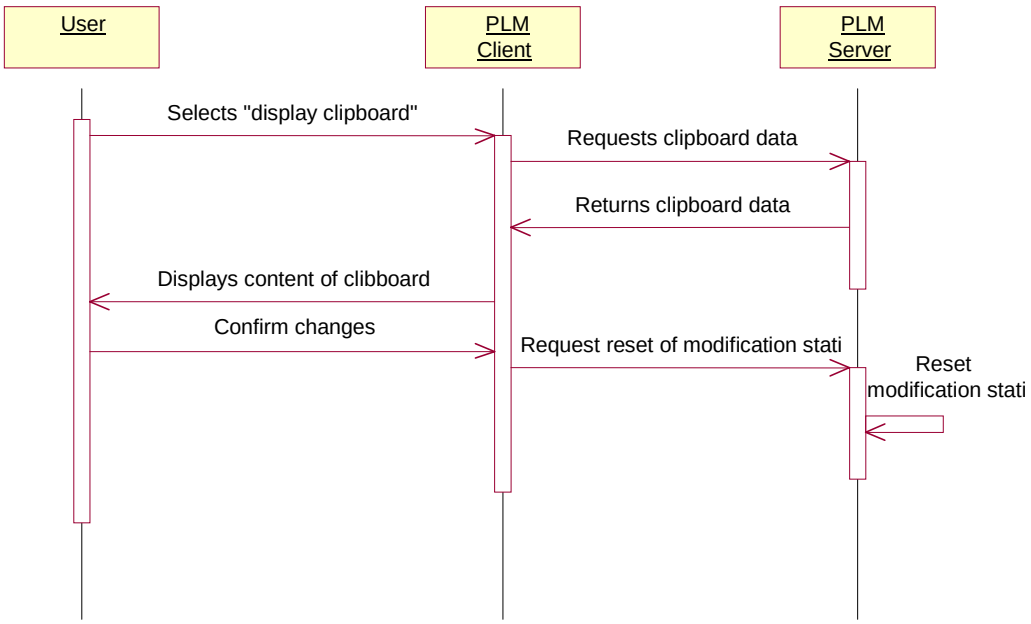


**Figure 7.10 - Display Content o Subscription list and confirm changes Process Flow**

### 7.2.17.5 Process start and end states

Start state / precondition S1:

A specific engineering project is defined, which itself defines certain items of product data (e.g. assemblies, parts, documents), that will be subject to change or creation during the project life time. These items are identified by identifiers.

- The user is correctly logged in and authorized to access the requested information.

End state / post condition E1 (Success):

- The process results in a virtual container (see use case “Search in design space”) containing all the objects in the subscription list.
- Objects modified since the last look on the subscription list are displayed emphasized. Deleted objects are displayed in a different style.
- After confirmation, the modification status of the objects is reset and in the case of deleted objects in the PLM system, they are also deleted from the subscription list.

### 7.2.17.6 Constraints and assertions

Currently none are defined.

#### 7.2.17.7 Relevant data

- Product meta data
- Work management data

### 7.2.18 Change content of subscription list

#### 7.2.18.1 Process purpose

The idea of the subscription list is, that the user needs a sort of folder in which he can collect objects. The purpose of the Subscription lists is to collect objects for which the change notification should be provided. The modification of the objects in this subscription lists is tracked and the user will be notified if such a modification takes place. The user should be able to change the content of his subscription list. The subscription list contains all objects the user wants to be notified when changes are applied to them.

#### 7.2.18.2 Partner/actor role descriptions

**Table 7.16 - Roles for changing content of subscription list**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.18.3 Process definition

1. The user selects objects in the subscription list and wants the PLM system to delete the objects from the subscription list.
2. The user selects objects in the PLM client and wants the PLM system to link those objects into the subscription list.

The following requirements are defined:

- The user has got a subscription list in the PLM system.
- For use case a), the content of the subscription list with the objects to delete have to be displayed.0
- For use case b), the objects to add have to be displayed in the client.

### 7.2.18.4 Process diagram

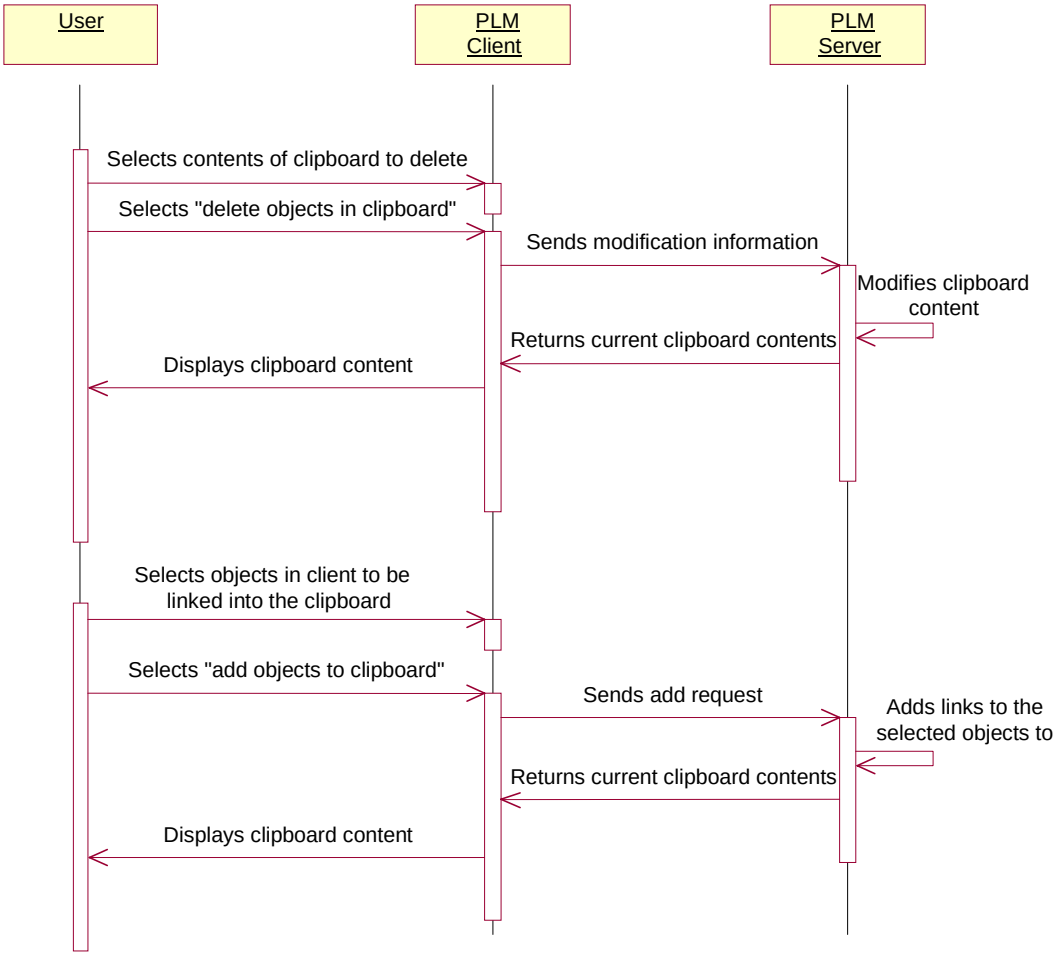


**Figure 7.11 - Change content of subscription list process flow**

### 7.2.18.5 Process start and end states

Start state / precondition S1 (use case a):

A specific engineering project is defined, which itself defines certain items of product data (e.g. assemblies, parts, documents), that will be subject to change or creation during the project life time. These items are identified by identifiers.

- The user is correctly logged in and authorized to access the requested information.
- The content of the subscription list is being displayed in the client.

Start state / precondition E2 (use case b):

A specific engineering project is defined, which itself defines certain items of product data (e.g., assemblies, parts, documents), that will be subject to change or creation during the project life time. These items are identified by identifiers.

- The user is correctly logged in and authorized to access the requested information.
- Product data is displayed.

End state / post condition E1 and E2 (Success):

- The process results in an updated view to the subscription list.

#### 7.2.18.6 Constraints and assertions

The user must own a subscription list.

#### 7.2.18.7 Relevant data

Product meta data

### 7.2.19 Product Class Identification

#### 7.2.19.1 Process purpose

Identification of a top level product_class to enable browsing of an abstract product structure.

#### 7.2.19.2 Partner/actor role descriptions

**Table 7.17 -Roles for product class identification**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.19.3 Process definition

This use case defines the process of identifying the start node of an abstract product structure in a PLM server. The end state / post condition of the use case is the precondition of the use case “Browsing of an abstract product structure.”

The process steps are:

- The user enters an ID or Wild Card.
- PLM server receives ID or Wild Card and triggers search in PLM System.
- Exception: The PLM server does not respond.

- PLM System executes query in its database

-> Exception: Database is not available, no data found, user is not authorized to access the data, etc.

- PLM server returns a list of product_class and product_component nodes.
- PLM client displays the resulting product_class nodes. If the list has only one member, it shall be displayed as the root node of a tree. If the list contains more than one node, then the result should be displayed as a list from which the user may select one node that is then displayed as the root node of a tree.

Remark: according to the AP214 CC8 Recommended Practices, each product_class is associated to one instance of product_component (with relation_type='realization') having the same attribute values. From this instance of product_component (not displayed within the client), the abstract product structure may be traversed (ProductStructureQuery).

#### 7.2.19.4 Process diagram

At the moment no process diagram is provided.

#### 7.2.19.5 Process start and end states

Start state / precondition S1:

- The user is correctly logged in and authorized to access the requested information.
- The service is available.
- The user enters an Id or Wild Card.

End state / post condition E1 (Success):

- The list of resulting nodes is displayed as described above.

End state / post condition E2 (Failure):

- The process results in a failure message.

#### 7.2.19.6 Constraints and assertions

None.

#### 7.2.19.7 Relevant data

- Product_class information

### 7.2.20 Browsing of Abstract Product Structures

#### 7.2.20.1 Process purpose

This process allows a user starting with an identified product_class, product_component, or alternative_solution to get information on the subcomponents of an abstract product structure (product_component or item_instance).

#### 7.2.20.2 Partner/actor role descriptions

**Table 7.18 - Roles for browsing of abstract product structures**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.20.3 Process definition

The process steps are:

- The PLM client evaluates if the product structure information is already obtained then it is directly displayed in a table.
- The PLM client sends a query for a substructure of product_class, product_component, or alternative_solution specified by the user to the PLM server.
- For each product structure node in the scope of the query the PLM server.
- Checks the authorization regarding the requested data .

-> Exception: Access denied

- Collects requested data within the PLM server.
- PLM server sends data to the PLM client.
- PLM client displays the resulting nodes within the structure. The kind of relationship (e.g product_structure_relationship of kind “decomposition” or “realization”) and child node (product_component or item_instance) should be displayed within the PLM client.

Remark: only one level of the product structure is retrieved at a time.

Remark: only product_structure_relationships from product_component to product_component from alternative_solution to item_instance and from alternative_solution to product_component are supported.

Remark: all the subtypes of item_instance are supported (single, quantified and selected). selected_instance is used in the case of a quantity 'as needed': selected_instance.selection_quantity refers to an instance of value_limit with limit=0 and limit_qualifier='minimum'.

Remark: this functionality is also available on item_version nodes if they are handled both as part (for their usage) as well as product_component (having an own abstract product structure). In this case, the function handles the item_version just as if it was a product_component.

#### 7.2.20.4 Process diagram

At the moment no diagram is provided.

#### 7.2.20.5 Process start and end states

Start state / precondition S1:

- The user is correctly logged in and authorized to access the requested information.
- The service is available.
- The user enters an Id.

End state / post condition E1 (Success):

- The list of resulting of the resulting nodes is displayed as described above.

End state / post condition E2 (Failure):

- The process results in a failure message.

#### 7.2.20.6 Constraints and assertions

None.

#### 7.2.20.7 Relevant data

- Product_structure_relationships, Product_components, Alternative_solutions, Item_instances

### 7.2.21 Browsing of Alternative Solutions within an Abstract Product Structure

#### 7.2.21.1 Process purpose

This process allows a user starting with an identified product_component (or alternative_solution) to get information on the (sub-)alternative solutions of an abstract product structure.

#### 7.2.21.2 Partner/actor role descriptions

**Table 7.19 - Roles for browsing of alternate solutions within an abstract product structure**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.21.3 Process definition

The process steps are:

- The PLM client evaluates if the alternative_solutions are already obtained, then it is directly displayed in a table.

- The PLM client sends a query for the alternative solutions of a product_component (or alternative_solution) specified by the user to the PLM server.
- For each alternative solution node in the scope of the query the PLM server.
- Checks the authorization regarding the requested data.

-> Exception: Access denied

- Collects requested data within the PLM server.
- PLM server sends data to the PLM client.
- PLM client displays the resulting nodes within the structure. The kind of child node (alternative_solution, technical_solution, final_solution, supplier_solution) should be displayed within the PLM client.

#### 7.2.21.4 Process diagram

At the moment no diagram is provided.

#### 7.2.21.5 Process start and end states

Start state / precondition S1:

- The user is correctly logged in and authorized to access the requested information.
- The service is available.
- The user enters an Id.

End state / post condition E1 (Success):

- The list of resulting of the resulting nodes is displayed as described above.

End state / post condition E2 (Failure):

- The process results in a failure message.

#### 7.2.21.6 Constraints and assertions

None.

#### 7.2.21.7 Relevant data

- Product_structure_relationships, Product_components

### 7.2.22 Retrieve Configuration Data within an Abstract Product Structure

#### 7.2.22.1 Process purpose

This process allows a user starting with an identified alternative_solution or item_instance to get information on the configuration of an abstract product structure.

#### 7.2.22.2 Partner/actor role descriptions

**Table 7.20 - Roles for retrieving configuration data within an abstract product structure**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.22.3 Process definition

The process steps are:

- The PLM client evaluates if configuration information is already obtained, then it is directly displayed in a table.
- The PLM client sends a query for the configuration[s] of an alternative_solution or item_instance specified by the user to the PLM server.
- For [each] configuration node in the scope of the query the PLM server.
- Checks the authorization regarding the requested data.

-> Exception: Access denied

- Collects requested data within the PLM server.
- PLM server sends data to the PLM client.
- PLM client displays the resulting nodes within the structure. The associated Specification referenced through Configuration and Class_specification_association should be displayed within the PLM client as a property of the configuration.

Remark: currently, configuration may be only displayed on alternative_solution and item_instance, but not on product_component and product_function.

Remark: for complexity reason the specification_expression corresponding to the logical rule stored within the legacy system is mapped to a single string and mapped to a pseudo-Specification.id. This specification is directly referenced by the Class_specification_association. The category of this specification has id=/DUMMY.

Remark: the product_class referenced by the class_specification_association will not be displayed to the PLM client, since it is either derived from the root node of the abstract product structure, or is project independent (for example in the case on configured assembly structures) and would have to be instantiated with a product_class of kind 'enterprise.'

Remark: if the usage of a part or product_component is not configured (i.e. the associated logical rule is empty), this function will give no results.

#### 7.2.22.4 Process diagram

At the moment no diagram is provided.

#### 7.2.22.5 Process start and end states

Start state / precondition S1:

- The user is correctly logged in and authorized to access the requested information.
- The service is available.
- The user enters an Id.

End state / post condition E1 (Success):

- The list of resulting of the resulting nodes is displayed as described above.

End state / post condition E2 (Failure):

- The process results in a failure message.

#### 7.2.22.6 Constraints and assertions

None.

#### 7.2.22.7 Relevant data

- Alternative_solution, Item_instance, Configuration, Product_class, Class_specification_association, Specification, Specification_category, see Section 8.3 and Section 8.10.

### 7.2.23 Viewing of Change Management Information

#### 7.2.23.1 Process purpose

Browsing through a product structure the user is able to see the assigned change management information.

#### 7.2.23.2 Partner/actor role descriptions

**Table 7.21 - Roles for viewing of change management information**

| Role name | Role description | Role type |
|---|---|---|
| User | Party, that requests PDM data. This could be a person, who interacts with the PLM client, or a system, that triggers the PLM client. | Person / System |
| PLM client | System, that provides the communication between user and PLM server. | System |
| PLM server | System, that provides the relevant PDM data. This is usually a company's PLM system that acts as a server. | System |

#### 7.2.23.3 Process definition

The process steps are:

- The user selects a node (product_class, product_component, item_version) within the PLM client.

- The PLM client evaluates if work management information is already obtained then it is directly displayed in a table.
- If work management information is not obtained the PLM client sends a query for this node to the PLM server.
- PLM System executes query in its database.

-> Exception: Database is not available, no data found, user is not authorized to access the data, etc.

- PLM server sends obtained work management data to the PLM client.
- PLM client displays the resulting data in a table.

Remark: according to the CC8 Recommended Practices, the effectivity references an event_reference, which references again an activity. Effectivity_assignment.effective_element and Activity_Element.element both reference the product_class, product_component or item_version node.

Remark: other object nodes are not supported at this time.

#### 7.2.23.4 Process diagram

At the moment no diagram is provided.

#### 7.2.23.5 Process start and end states

Start state / precondition S1:

- The user is correctly logged in and authorized to access the requested information.
- The service is available.
- The user selects a node of kind product_class, product_component or item_version in the tree view.

End state / post condition E1 (Success):

- The resulting information is displayed as described above.

End state / post condition E2 (Failure):

- The process results in a failure message.

#### 7.2.23.6 Constraints and assertions

None.

#### 7.2.23.7 Relevant data

- Activity, Activity_element, Effectivity, Effectivity_assignment, Event_reference, see Section 8.10 and Section 8.11.

### 7.2.24 ECM participant proposal for a change

#### 7.2.24.1 Owner of the use case

This use case was defined by the joint ECM Project Group of the ProSTEP iViP association and the VDA.

#### 7.2.24.2 Process purpose

The ECM participant requests a new engineering change request and is not involved in subsequent processing of this request. This use case is used if the participant identifies the potential need for an engineering change but is not directly affected by the engineering change itself (for example a change to a part which is the responsibility of a partner/supplier). In this use case, an automotive OEM is typically the ECM coordinator and a supplier is typically the ECM participant.

#### 7.2.24.3 Partner role descriptions

**Table 7.22 - Roles for the use case ECM participant proposal for a change**

| Role name | Role description | Role type |
|---|---|---|
| ECM coordinator | Owner of the engineering change request (e.g., an automotive OEM). | organization |
| ECM participant | Party involved in the elaboration of the engineering change request (e.g., a part supplier). | organization |
| ECM server | ECM system, that provides the relevant ECM data. It is usually the ECM system of the ECM coordinator that acts as a server. | system |
| ECM client | System of the ECM participant, that provides the communication between ECM participant and ECM server of the ECM coordinator. | system |

#### 7.2.24.4 Process definition

The ECM participant uses the ECM client to send a message (Request_initial_ECR) to the ECM coordinator to request him to create a new engineering change request. The ECM server of the ECM coordinator first sends a response message (Respond_initial_ECR) to the request. This message contains the coordinator's and the participant's request number (counterparts). The engineering change request is subsequently handled under the responsibility of the ECM coordinator. The ECM client of the ECM participant then receives the message notifying him whether the engineering change request has been created (Notify_initial_ECR_accepted) or rejected (Notify_initial_ECR_rejected). If the engineering change request has been created, only one further message (Notify_ECR_decided) is received by the ECM client, notifying the ECM participant whether the ECM coordinator has approved, rejected, or canceled the request.

### 7.2.24.5 Process diagram

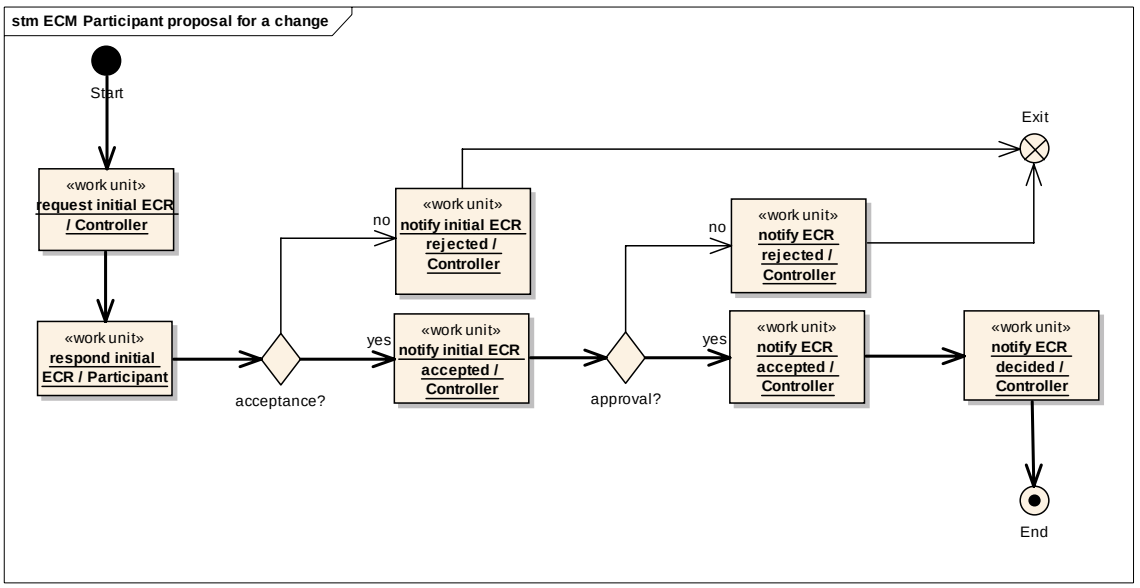


**Figure 7.12 - Illustration of the process definition of the use case “ECM Participant proposal for a change”**

### 7.2.24.6 Process start and end states

Start state / precondition:

- The ECM participant has elaborated an engineering change request.

End state / post condition:

- The ECM coordinator has rejected the engineering change request of the ECM participant (failure).
- The ECM coordinator has further elaborated/detailed and approved the engineering change request of the ECM participant (success).
- The ECM coordinator has further elaborated/detailed and disapproved the engineering change request of the ECM participant (failure).

### 7.2.24.7 Constraints and assertions

For constraints and assertions of this use case, see [x].

### 7.2.24.8 Relevant data

- Work management data, see Section 8.11.

### 7.2.24.9 Topics under discussion / Remarks

At the moment, there are no topics under discussion.

### 7.2.25 ECM participant comments

#### 7.2.25.1 Owner of the use case

This use case was defined by the joint ECM Project Group of the ProSTEP iViP association and the VDA.

#### 7.2.25.2 Process purpose

In this use case, the ECM participant is requested to provide a technical and/or commercial evaluation and comment of an ECR, but the ECM participant was not involved in the preceding technical analysis and detailing phase.

In this use case, an OEM is typically the coordinator of the engineering change request. The engineering change request is primarily within his responsibility. For sets of parts developed or manufactured externally, the supplier is incorporated in the role of an ECM participant and provides comments for the parts which affect him or for which he is responsible for.

#### 7.2.25.3 Partner role descriptions

**Table 7.23 - Roles for the use case “ECM participant comments”**

| Role name | Role description | Role type |
|---|---|---|
| ECM coordinator | Owner of the engineering change request (e.g., an automotive OEM). | organization |
| ECM participant | Party involved in the elaboration of the engineering change request (e.g., a module supplier). | organization |
| ECM server | ECM system, that provides the relevant ECM data. It is usually the ECM system of the ECM coordinator that acts as a server. | system |
| ECM client | System of the ECM participant, that provides the communication between ECM participant and ECM server of the ECM coordinator. | system |

#### 7.2.25.4 Process definition

Simple case: The ECM coordinator requests that the ECM participant provide a comment of the engineering change request (Request_ECR_comments). The ECM participant provides the requested information (Respond_ECR_comments) and receives a message notifying him whether the request was approved or rejected by the coordinator (Notify_ECR_decided) on completion of the entire engineering change request.

Options: If the need for an engineering change is first identified by an ECM participant, he can send the request to create an engineering change request to the coordinator (Request_initial_ECR). The ECM participant first receives a confirmation that his request has been received under the initial engineering change request number of the coordinator (Respond_initial_ECR). Later on, the ECM coordinator sends the message Notify_initial_ECR_rejected if the request was rejected, i.e., if no engineering change request was created, or the message Notify_initial_ECR_accepted if a regular engineering change request was created. The subsequent procedure is the same as for the simple case.

In addition to the simple case or to the aforementioned option, the ECM coordinator can also request that the ECM participant provide a confirmation comment following receipt of the comments (Request_ECR_confirmation_comments). Depending on the agreement made, the ECM participant provides his feedback and complete comments on the entire engineering change request or on missing or incorrect information and provides any additional information (Respond_ECR_confirmation_comments).

### 7.2.25.5 Process diagram

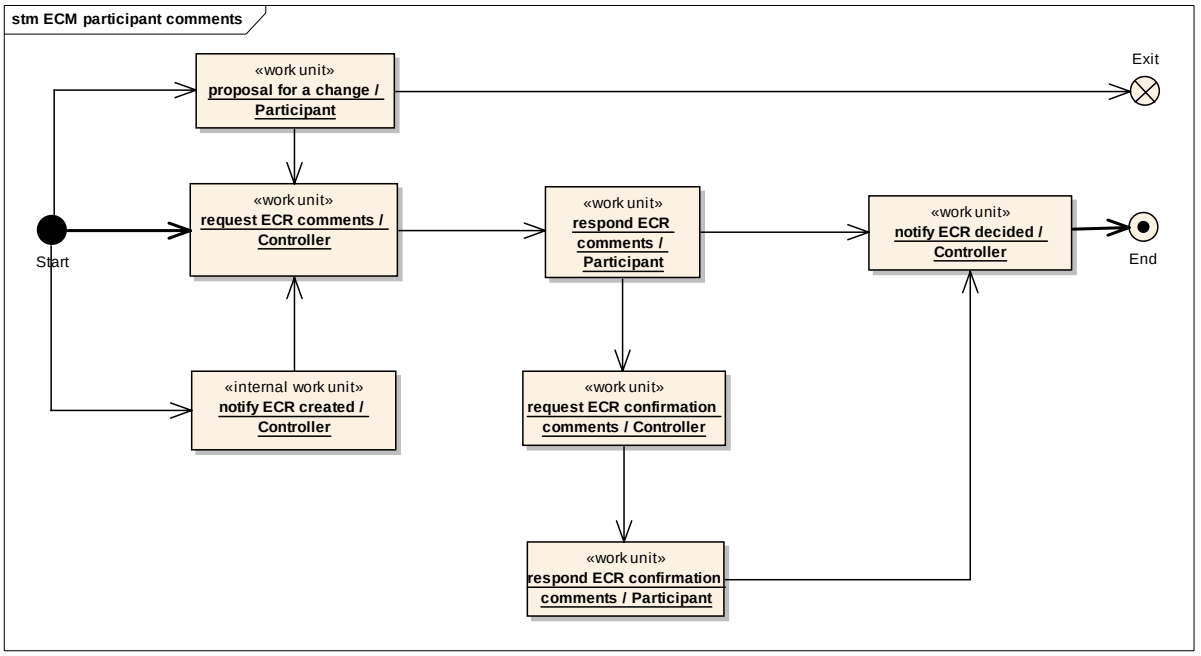


**Figure 7.13 - Illustration of the process definition of the use case “ECM participant comments”**

### 7.2.25.6 Process start and end states

Start state / precondition:

- The ECM coordinator requests comments about an ECR from an ECR participant (simple case).
- The ECM participant has elaborated an engineering change request.

End state / post condition:

- The ECM coordinator has rejected the engineering change request of the ECM participant (failure).
- The ECM coordinator has further elaborated/detailed and approved the engineering change request (success).
- The ECM coordinator has further elaborated/detailed and disapproved the engineering change (failure).

### 7.2.25.7 Constraints and assertions

For constraints and assertions of this use case, see [VDA].

#### 7.2.25.8 Relevant data

- Work management data, see Section 8.11.

#### 7.2.25.9 Topics under discussion / Remarks

At the moment, there are no topics under discussion.

### 7.2.26 ECM participant approval

#### 7.2.26.1 Owner of the use case

This use case was defined by the joint ECM Project Group of the ProSTEP iViP association and the VDA.

#### 7.2.26.2 Process purpose

This use case comes into play when the supplier has full responsibility for development of certain sets of parts of the OEM and these are to be changed on the initiative of the partner, or when the supplier acts as the prime contractor in the vehicle project.

In this case, the supplier takes on the role of the ECM coordinator and the OEM takes on the role of the ECM participant. In this use case, the need to make an engineering change is identified only by the ECM coordinator and communication between the ECM coordinator and the ECM participant takes place solely in the approval phase of the ECR process.

#### 7.2.26.3 Partner role descriptions

**Table 7.24 - Roles for the use case “ECM participant approval”**

| Role name | Role description | Role type |
|---|---|---|
| ECM coordinator | Owner of the engineering change request (e.g., a prime contractor). | organization |
| ECM participant | Party involved in the elaboration of the engineering change request (e.g., an automotive OEM). | organization |
| ECM server | ECM system, that provides the relevant ECM data. It is usually the ECM system of the ECM coordinator that acts as a server. | system |
| ECM client | System of the ECM participant, that provides the communication between ECM participant and ECM server of the ECM coordinator. | system |

#### 7.2.26.4 Process definition

In this use case, the ECM coordinator issues an engineering change request internally, elaborates the engineering change request in detail, provide comments on the proposed changes from his point of view and sends this detailed engineering change request, which is complete from his point of view, to the ECM participant, requesting him to check the technical completeness of the request (Request_ECR_completeness_check message).

The ECM participant first sends back to the ECM coordinator the engineering change request number he is using internally to track the engineering change request in the Notify_ECR_id message, together with the ECM coordinator's engineering change request number. The ECM participant can reject the request at this early stage. This is done using the "reject" flag. The ECM coordinator must respond by sending a Notify_ECR_decided message.

If the ECM participant did not reject the engineering change request early, the ECM participant sends his response to the ECM coordinator's request for a completeness check in the Respond_ECR_completeness_check message.

- If the ECM coordinator's engineering change request is incomplete from the point of view of the ECM participant, the ECM participant informs the coordinator using the flag "is_not_complete" and includes the set of parts which are missing in his opinion and/or his comments. The ECM coordinator must then send a further Request_ECR_completeness_check message.
- If, on the other hand, the engineering change request is complete, the ECM participant informs the ECM coordinator using the flag "is_complete".

As soon as the ECM coordinator has received a positive response from the ECM participant with respect to the completeness of the engineering change request, he can send the final engineering change request to the participant using the Request_ECR_acceptance message, requesting the ECM participant to approve the engineering change request.

The ECM participant sends his response to the ECM coordinator using the Respond_ECR_acceptance message. This response takes the form of approval or rejection from the point of view of the ECM participant.

Finally, the ECM coordinator informs the ECM participant of the final decision regarding the engineering change request in the Notify_ECR_decided message. This message is also sent if the ECM participant has rejected the engineering change request early using the "reject" flag in the Request_ECR_completeness_check message.

### 7.2.26.5 Process diagram

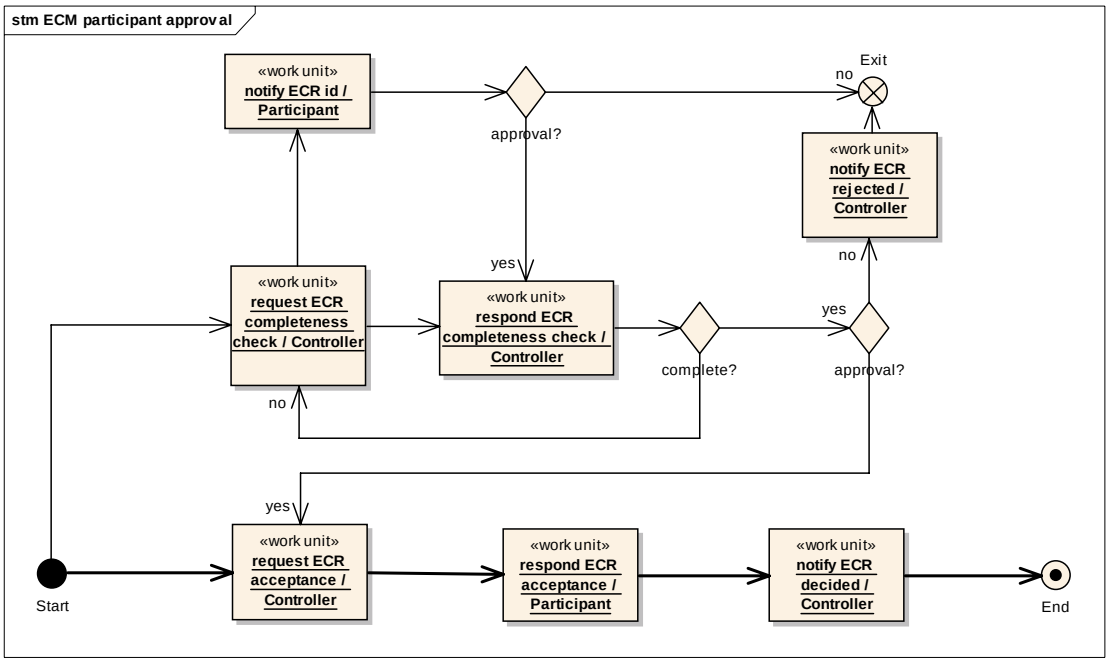


**Figure 7.14 - Illustration of the process definition of the use case "ECM participant approval"**

#### 7.2.26.6 Process start and end states

Start state / precondition:

- The ECM coordinator sends the Request_ECR_completness_check message to the ECM participant

End state / post condition:

- The ECM coordinator has finally approved the engineering change request (success).
- The ECM coordinator has finally disapproved the engineering change (failure).

#### 7.2.26.7 Constraints and assertions

For constraints and assertions of this use case, see [x].

#### 7.2.26.8 Relevant data

- Work management data

#### 7.2.26.9 Topics under discussion / Remarks

At the moment, there are no topics under discussion.

### 7.2.27 ECM participant detailing and comments

#### 7.2.27.1 Owner of the use case

This use case was defined by the joint ECM Project Group of the ProSTEP iViP association and the VDA.

#### 7.2.27.2 Process purpose

This use case describes a joint engineering change process where the ECM coordinator and ECM participant work closely together throughout all phases of the process. A typical application is the development and production of complete systems or entire vehicles in a cooperative venture. Typically, the OEM takes on the role of the ECM coordinator and is cooperation partner is in the role of the ECM participant.

#### 7.2.27.3 Partner role descriptions

**Table 7.25 - Roles for the use case “ECM participant detailing and comments”**

| Role name | Role description | Role type |
|---|---|---|
| ECM coordinator | Owner of the engineering change request (e.g., an automotive OEM). | organization |
| ECM participant | Party involved in the elaboration of the engineering change request (e.g., a system supplier or a prime contractor). | organization |

**Table 7.25 - Roles for the use case “ECM participant detailing and comments”**

| | | |
|---|---|---|
| ECM server | ECM system, that provides the relevant ECM data. It is usually the ECM system of the ECM coordinator that acts as a server. | system |
| ECM client | System of the ECM participant, that provides the communication between ECM participant and ECM server of the ECM coordinator. | system |

### 7.2.27.4 Process definition

Either the ECM coordinator (Notify_ECR_creation) or the ECM participant (Request_initial_ECR) requests the creation of an engineering change request. In the technical analysis phase, the ECM coordinator obtains information about particular objects (parts and documents) which affect the ECM participant (Request_ECR_details). The ECM participant can also add further objects to the engineering change request (Respond_ECR_details). If required, the ECM coordinator can include the ECM participant in the commenting phase. He can request a comment on individual affected objects (e.g., the costs for a part) or on the entire engineering change request including any comments already received from another ECM participant (Request_ECR_comments). Finally, the ECM coordinator informs the ECM participant of the decision made with respect to the engineering change request.

### 7.2.27.5 Process diagram

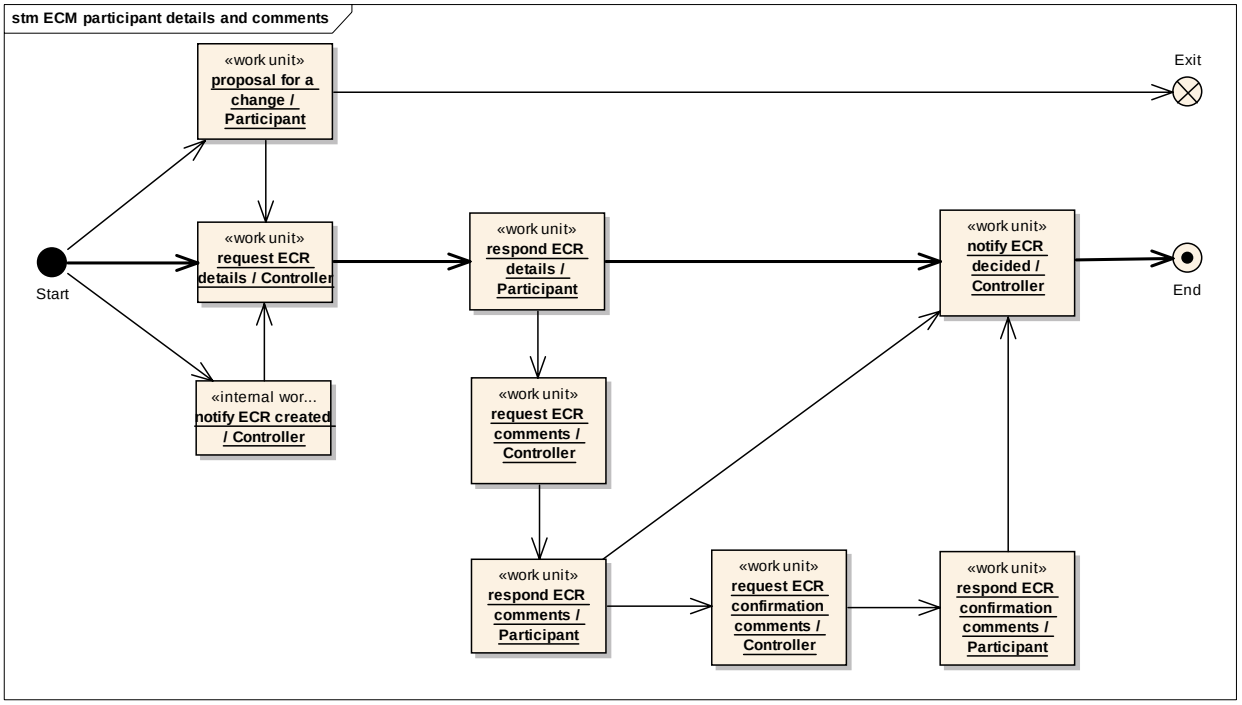


**Figure 7.15 - Illustration of the process definition of the use case “ECM participant detailing and comments”**

#### 7.2.27.6 Process start and end states

Start state / precondition:

- The ECM coordinator creates an ECR
- The ECM participant has elaborated an engineering change request.

End state / post condition:

- The ECM coordinator has rejected the engineering change request of the ECM participant (failure)
- The ECM coordinator has further elaborated/detailed and approved the engineering change request (success)
- The ECM coordinator has further elaborated/detailed and disapproved the engineering change (failure)

#### 7.2.27.7 Constraints and assertions

For constraints and assertions of this use case, see [VDA].

#### 7.2.27.8 Relevant data

- Work management data, see Section 8.11.

#### 7.2.27.9 Topics under discussion / Remarks

At the moment, there are no topics under discussion.

# 8 Informational Viewpoint PIM (normative)

In this section the PIM in the informational viewpoint of the PLM Services 2.0 is defined. It is derived from the STEP AP214 CC21 ARM model.

Additionally, some new classes were created and put into a package called "PLM_Base". This package realizes two modeling concepts of the PIM. Firstly, it introduces the concept of identifying instances by an unique identifier. This identifier must be unique throughout a session as defined by the computational model in Chapter 9. Secondly, it defines a container concept to establish a correct handling of the data passed to and from the computational model.

All classes and interfaces are listed with their packages, base classes, attributes, compositions, and associations. Additionally the classes and their members are described textually. The text of all descriptions (except for the "PLM Base") are reproduced from ISO 10303-214 with permission of ISO. The copyright remains with ISO.

The PIM Informational Model has the following package hierarchy:

Package PLM_services

- Package PLM_base
- Package Part_identification
  This package includes the primary objects used for product data management. This subset provides the capability to represent product management information. It includes information about items that are either raw materials, parts, or tools, about versions and views of items. A part may represent one of a variety of physical entities used in discrete manufacturing; including raw material, semi-finished parts, assemblies, instruction manuals, kits, manufacturing by-products, and products. The manufacturing industry is de-fined by the design, production, and sales of parts, and almost every business activity in some way works with data that describes parts.
- Package Part_structure
  Base of this package is the group of objects that define the bill of material relationships be-tween items for discrete manufacturing.

  A part is not defined by a single object with a set of attributes, but a collection of objects and relationships, each describing different aspects of the part. For example, a part definition may consist of several engineering attributes, links to suppliers of the part, references to CAD drawings describing the parts geometry, and a list of components used to assemble the part. These different pieces of the part definition will be referred to as part data objects.This subset supports explicit hierarchical product structures representing assemblies and the constituents of those assemblies. This explicit part structure corresponds to the traditional engineering and manufacturing bill of material indentured parts list.
- Package Document_and_file_management
  The scope of this package is the handling of electronic documents comprising one or more files and track documents that are not actively managed by the PLM system.

  External files represent a simple external reference to a named file. An external file is not managed independently by the system - there is usually no revision control or any representation definitions of external files. Version identification may optionally be associated with an external file, but this is for information only and is not used for managed revision control.
- If a file is under configuration control, it should be represented as a constituent of a document definition view/representation. In this case it is actually the managed document that is under direct configuration control, the file is in this way indirectly under configuration control. A change to the file results in a change to the managed document (i.e., a

new version) - the changed file would be mapped as a constituent of a view/representation definition of the new document version. A simple external reference alone is not configuration controlled; it is just an external file reference to product data. Documents may be associated with product data in a specified role, to represent some relationship between a document and other elements of product data. Constraints may also be specified on this association, in order to distinguish an applicable portion of an entire document or file in the association.

- Package Shape_definition and_transformation
  The scope of this subset provides the capability to associate items with shape or to identify aspects of the shape. It allows also to distinguish between geometric elements used as auxiliary elements and geometric elements that describe product data. Additionally, it contains the capability of an empty geometric model with only a geometric element for placement purposes and an unconstrained three-dimensional geometric model that may contain any geometric data elements.

  This subset allows linking geometric structures that result from relating different shape representations with associated product structure when applicable, i.e., when the geometric structure directly corresponds to the assembly structure.

  Two alternatives for the implementation of geometric structures related to assembly structures are recommended:

  - The assembly is described with the components built in. With this approach the shape of the component is mapped into the shape of the assembly via mapped_item. The basic idea of the mapped_item is: an item will become part of another item. The assembly component geometry is used as a template in the assembly geometry.
  - The components of an assembly are described together with the construction history. This approach uses the representation_relationship_with_transformation. The transformation describes the relation between different workspaces.

- The usage of both alternatives is considered reasonable, because both mechanisms make sense even in mixed combinations. With regard to the transformations in the context of assembly, a part is in principle incorporated in the assembly only by rigid motion (i.e., translation and/or rotation) excluding mirroring and scaling.

- Package Classification
  A simple basic type of classification of products in STEP works by assigning categories to product data items. These categories are identified by name labels that define the related classification. This type of classification is referred to as specific classification. A specific_item_classification_hierarchy is used to build up hierarchical structures of specific_item_classification.

- Package Properties
  The scope of this subset allows specifying properties associated with parts. A property is the definition of a special quality and may reflect physics or arbitrary, user defined measurements. A general pattern for instantiating property information is in this subset. A number of pre-defined property type names are also proposed for use when appropriate.

  A special case of part properties is that of the part shape property - a representation of the geometrical shape model of the part, which are described in section 2.3.4.

- Package Alias_identification
  An alias identification is a mechanism to associate an object with an additional identifier that is used to identify the object of interest in a different context, either in another organization, or in some other context. The alias identification mechanism shall not be used to alias sup-plied parts.

  The scope of the alias identification shall be specified either by the description of the associated identification_role or - if the scope is defined by an organization - with help of an applied_organization_assignment. The scope of an alias defines the context in which the id specified via applied_identification_assignment.assigned_id overrides the original id. A scenario might be that an object has an id in the context of the organization assigned in the role 'id owner' as a primary id and other ids defined via aliases that are valid in the context of some other organizations.

- Package Authorization
  The scope of this subset represents organizations and people in organizations as they per-form functions related to other product data and data relationships. A person in this scope must exist in the context of some organization. An organization or a person in an organization is then associated with the data or data relationship in some role indicating the function being performed. Both people and organizations may have addresses associated with them.

  Approving in this scope is accomplished by establishing an approval entity and relating it to some construct through an applied_approval_assignment. The applied_approval_assignment entity may have a role associated with it through the entity role_association and its related object_role entity to indicate the reason/role of this approval related to the particular element of product data.

  Approval may be represented as a simple basic approval, or it may represent a more complex approval cycle involving multiple provers, on different dates/times, and possibly with different status values.

- Package Configuration_management
  The purpose of this package from the STEP AP214 is meeting the requirements of enterprises that offer many possible configurations of their products for sale. In most cases, the different configurations of a product differ from each other in only minor ways. Configuration identification is the identification of product concepts and their associated configurations, the composition of which is to be managed. If a configuration of a product concept is implemented by a certain design, i.e. a particular part version, this version can be associated with the configuration and managed using configuration effectivity. Because this model is based on the configuration management model defined in STEP AP214, additional information and description of how to use the model can be found in the ARM model and other documentation on AP214.

- Package Change_and_work_management
  This package describes the process by which companies request, implement, and effect change to products, documents, components, assemblies, manufactured or purchased parts, processes, or even suppliers. This subset provides the capability to represent activity, project, and contract related information. Activities may be initiated by work requests and may be authorized by work orders. Activities may result in changes of models or of properties; such changes can also be represented.

- Package Process_planning
  This package provides the capability to represent process related data. This includes process plans, versions of process plans with version tracking, process operations and properties of processes. A process plan is decomposed into one or more occurrences of process operations. Process plans and process operations establish relationships among raw materials, in-process items, and final items, as well as the relationship between the items and the tools used to manufacture them. Additionally, the representation of the connection of parts in various kinds of mating is part of this package.

- Package Multi_language_support
  This package provides the capability to represent descriptive information about objects in different languages.

## 8.1 Package PLM_base

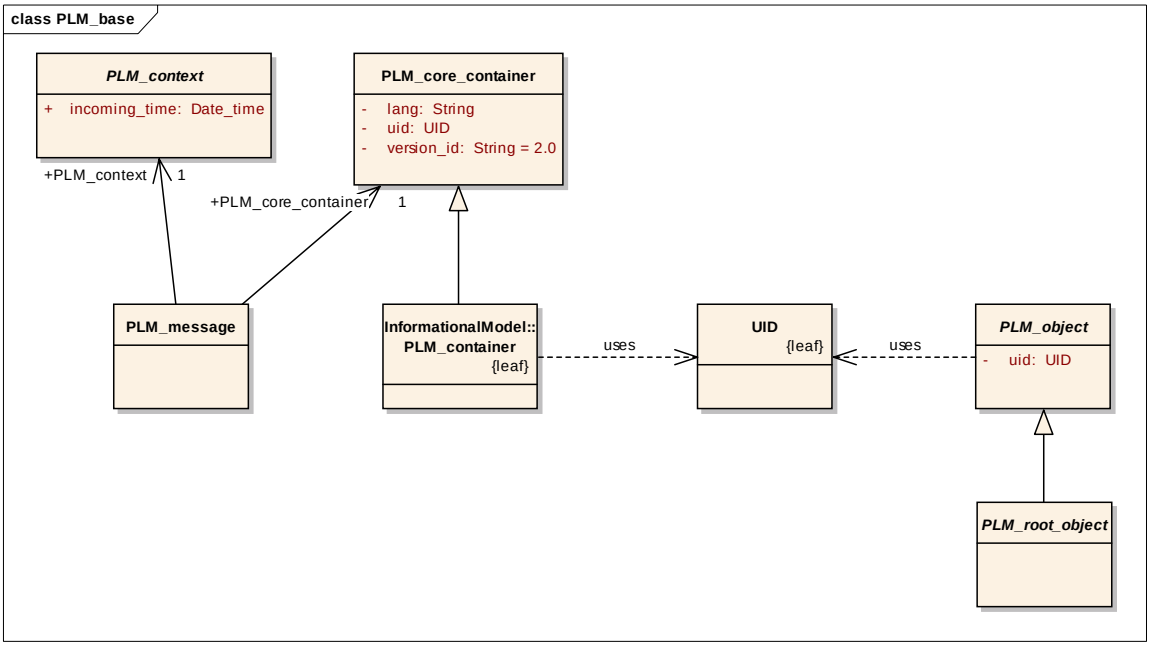


**Figure 8.16 - PLM base**

### 8.1.1 Classes

#### 8.1.1.1 Class PLM_core_container

All operations in this specification use the type PLM_core_container as input parameter type or return type when PLM data has to be transfered. So, the PLM_core_container serves as a container to transfer arbitrary PLM data.

***Base Class***

- none

***Attributes***

- lang:String[1]: Language identifying string
- uid:UID Unique identifier
- version_id:String[1]=2.0: Indicates the version number of the PLM Services specification this container is compliant to.

#### 8.1.1.2 Class PLM_container

The PLM_container class is introduced to ensure that data is only handled by the computational model in a valid way.

### *Base Class*

- PLM_core_container

### *Attributes*

- none

### *Compositions*

- accuracy : Accuracy [0..*]
- activity : Activity [0..*]
- activity_method : Activity_method [0..*]
- address : Address [0..*]
- application_context : Application_context [0..*]
- approval_status : Approval_status [0..*]
- axis2_placement_3d : Axis2_placement_3d [0..*]
- axis_placement : Axis_placement [0..*]
- cartesian_coordinate_space : Cartesian_coordinate_space (ABS) [0..*]
- cartesian_point : Cartesian_point [0..*]
- certification : Certification [0..*]
- classification_attribute : Classification_attribute [0..*]
- classification_system : Classification_system [0..*]
- complex_product : Complex_product (ABS) [0..*]
- contract : Contract [0..*]
- data_environment : Data_environment [0..*]
- date_time : Date_time [0..*]
- descriptive_specification : Descriptive_specification [0..*]
- design_constraint : Design_constraint [0..*]
- direction : Direction [0..*]
- document : Document [0..*]
- document_content_property : Document_content_property [0..*]
- document_creation_property : Document_creation_property [0..*]
- document_file : Document_file (ABS) [0..*]
- document_format_property : Document_format_property [0..*]
- document_location_property : Document_location_property [0..*]
- document_size_property : Document_size_property [0..*]
- document_type_property : Document_type_property [0..*]

- duration : Duration [0..*]
- effectivity : Effectivity [0..*]
- event_reference : Event_reference [0..*]
- external_library_reference : External_library_reference [0..*]
- general_classification : General_classification [0..*]
- geometric_model : Geometric_model [0..*]
- interval_of_time : Interval_of_time [0..*]
- item : Item [0..*]
- item_shape : Item_shape [0..*]
- language : Language [0..*]
- material : Material [0..*]
- model_property_value
- organization : Organization [0..*]
- person : Person [0..*]
- physical_instance : Physical_instance [0..*]
- plib_class_reference
- plib_property_reference
- process_operation_definition : Process_operation_definition [0..*]
- process_plan : Process_plan [0..*]
- product_class : Product_class [0..*]
- project : Project [0..*]
- property : Property (ABS) [0..*]
- property_value : Property_value (ABS) [0..*]
- rectangular_size : Rectangular_size [0..*]
- retention_period : Retention_period [0..*]
- security_level : Security_level [0..*]
- shape_dependent_property : Shape_dependent_property [0..*]
- specific_document_classification : Specific_document_classification [0..*]
- specific_item_classification : Specific_item_classification [0..*]
- specification : Specification [0..*]
- specification_category : Specification_category [0..*]
- specification_expression : Specification_expression [0..*]
- transformation : Transformation (ABS) [0..*]
- unit : Unit [0..*]

- work_order : Work_order [0..*]
- work_request : Work_request [0..*]

#### *Associations*

- none

### 8.1.1.3 Class PLM_context (ABS)

The PLM_context is an abstract base type for the context of a PLM_message, e.g., if the message is part of a workflow, the context describes the position of the message in the workflow.

#### *Base Class*

- none

#### *Attributes*

- incoming_time : Date_time [1]

  The attribute incoming_time is set by the PLM_message_connection implementation with the time when the containing PLM_message object was received as parameter of the operation write_messages.

#### *Compositions*

- none

#### *Associations*

- none

### 8.1.1.4 Class PLM_message

The PLM_message is a base type for a message exchange based client service communication and is used by the PLM_message_connection service in the Computational Viewpoint (see Chapter 9 ). The PLM_context describes the position of the message in a workflow and the PLM_core_container contains the user data of the message.

#### *Base Class*

- none

#### *Attributes*

- none

#### *Compositions*

- PLM_context : PLM_context [1]

  The PLM_context describes relevant data to identify the message, to associate it to a particular workflow and to determine the position of the message in a workflow.

- PLM_core_container : PLM_core_container [1]

  The PLM_core_container contains the pay load of the message.

#### *Associations*

- none

#### 8.1.1.5 Class PLM_object (ABS)

The abstract PLM_object class is introduced to provide a mechanism of binding a unique identifier to each PLM class instance. These identifiers must be valid and unique throughout a complete session defined by the computational model. After each session the identifiers may are invalid.

#### *Base Class*

- none

#### *Attributes*

- uid : UID [1] — The uid uniquely identifies an object throughout a complete session defined by the computational model. After each session the uid is invalid.

#### *Compositions*

- none

#### *Associations*

- none

#### 8.1.1.6 Class PLM_root_object (ABS)

The abstract class PLM_root_object is defined to distinguish between types which can be directly inserted into PLM_container instances and types which are contained in the container through PLM_root_object instances.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.1.2 Datatypes

Datatype Boolean

Datatype Double

Datatype Integer

Datatype String

Datatype UID

## 8.2 Package Part_identification

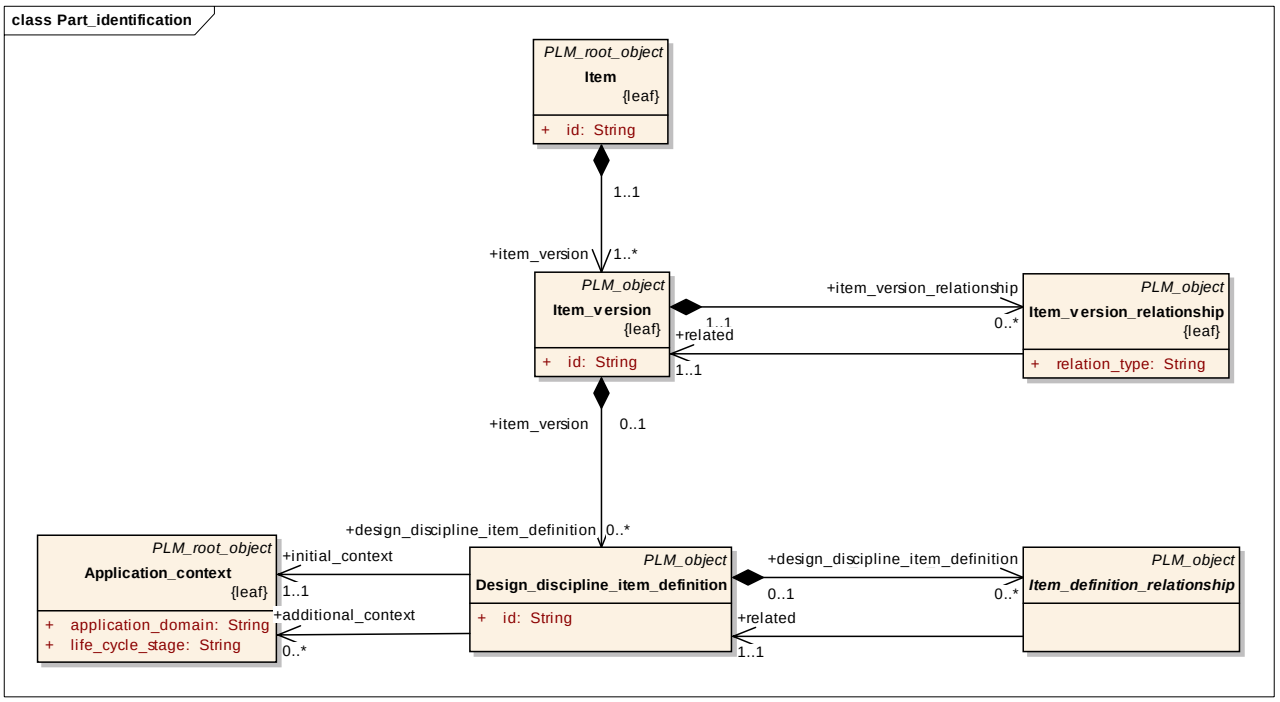


**Figure 8.17 - Part identification**

### 8.2.1 Classes

#### 8.2.1.1 Class Application_context

An Application_context is a shared universe of discourse.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- application_domain : String [1]The application_domain is the identification of the applications for which an object may be relevant. Where applicable the following values shall be used:
  - 'assembly study': The object may be relevant for an assembly study;
  - 'digital mock-up': The object may be relevant for digital mock-up;
  - 'electrical design': The object may be relevant for the electrical design;
  - 'mechanical design': The object may be relevant for the mechanical design;
  - 'preliminary design': The object may be relevant for the preliminary design;
  - 'process planning': The object may be relevant for the process planning.

- life_cycle_stage : String [1] The life_cycle_stage is the specification of the general stage in the product life cycle to which the concerned items belong. Where applicable the following values shall be used:
  - 'design': The concerned item belongs to the design phase of the life cycle;
  - 'manufacturing': The concerned item belongs to the manufacturing phase of the life cycle;
  - 'recycling': The concerned item belongs to the recycling phase of the life cycle.

#### *Compositions*

- description : String_select [0..1]
  The description specifies additional information about the Application_context.

#### *Associations*

- none

### 8.2.1.2 Class Design_discipline_item_definition

A Design_discipline_item_definition is a view of an Item_version. This view is relevant for the requirements of one or more life cycle stages and application domains and collects product data of the Item_version.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Design_discipline_item_definition.

#### *Compositions*

- item_instance : Item_instance (ABS) [0..*]
  The item_instance specifies the item_instance that is defined by this Design_discipline_item_definition.
- item_definition_relationship : Item_definition_relationship (ABS) [0..*]
  The item_definition_relationship specifies the Item_definition_relationship that relates the first of the two Design_discipline_item_definition objects.
- name : String_select [0..1] The name specifies the word or group of words used to refer to the Design_discipline_item_definition.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Design_discipline_item_definition.
- item_function_association : Item_function_association [0..*]
  The item_function_association specifies the Item_function_association which this Design_discipline_item_definition is associated with.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Design_discipline_item_definition.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

- Item_definition_instance_relationship : Item_definition_instance_relationship (ABS) [0..*]
  The Item_definition_instance_relationship specifies the Item_definition_instance_relationship which this Design_discipline_item_definition is part of. If the Design_discipline_item_definition is an Assembly_definition, the relationship shall be a Assembly_component_relationship. If the Design_discipline_item_definition is an Collection_definition, the relationship shall be a Collected_item_association.

#### *Associations*

- initial_context : Application_context [1]
  The initial_context specifies the Application_context in which this view of the Item_version has been designed primarily.
- additional_context : Application_context [0..*]
  The additional_context specifies the set of Application_context objects in which this view of the Item_version is also relevant. The additional_context shall not contain the Application_context that is referenced as the 'initial_context'.

### 8.2.1.3 Class Item

An Item is either a single object or a unit in a group of objects. It collects the information that is common to all versions of the object. An Item shall always be classified as 'part', 'tool', or 'raw material' using a Specific_item_classification. Additionally, if an Assembly_definition exists for at least one version of the Item, the Item shall be classified as being an 'assembly' using Specific_item_classification.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Item. For the id, an owner shall be specified by a Person_organization_assignment with role 'id owner'. The id shall be unique within the scope of the organization that is specified by the Person_organization_assignment with the role 'id owner'.

#### *Compositions*

- item_version : Item_version [1..*]
  The item_version specifies the Item_version that is associated with this Item.
- description : String_select [0..1]
  The description specifies additional information about the Item.
- name : String_select [1] The name specifies the word or group of words used to refer to the Item.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Item.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Item.

#### *Associations*

- none

#### 8.2.1.4 Class Item_definition_relationship (ABS)

An Item_definition_relationship is a relationship between two Design_discipline_item_definition objects.

***Base Class***

- PLM_object (ABS)

***Attributes***

- none

***Compositions***

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Item_definition_relationship.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

***Associations***

- related : Design_discipline_item_definition [1]
  The related specifies the second of the Design_discipline_item_definition objects that are part of the relationship.

#### 8.2.1.5 Class Item_version

An Item_version is a version of an Item and serves as the collector of the data characterizing a physically realizable object in various application contexts.

***Base Class***

- PLM_object (ABS)

***Attributes***

- id : String [1] The id specifies the identifier of the Item_version. The id shall be unique within the scope of the associated Item.

***Compositions***

- item_version_relationship : Item_version_relationship [0..*]
  The item_version_relationship specifies the item_version_relationship that relates the first of the two Item_version objects.
- description : String_select [0..1]The description specifies additional information about the Item_version.
- product_design : Product_design [0..1]
  The product_design specifies the Product_design for which the Item_version meets the requirements.
- design_discipline_item_definition : Design_discipline_item_definition [0..*]
  The design_discipline_item_definition specifies the Design_discipline_item_definition that is a view for this Item_version.

- alias_identification : Alias_identification [0..*]
  The alias_identification specifies the Alias_identification that is applied to this Item_version.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Item_version.

#### *Associations*

- none

### 8.2.1.6 Class Item_version_relationship

An Item_version_relationship is a relationship between two Item_version objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'derivation': The application object defines a deriving relationship where the related Item_version is based on the relating Item_version which is an earlier version of the same or of a different Item;
  - 'hierarchy': The application object defines a hierarchical relationship where the related Item_version is a subordinate version of the relating Item_version;
  - 'sequence': The application object defines a version sequence where the relating Item_version is the preceding version of the related Item_version that is the following version. For a given Item_version there shall be at most one Item_version_relationship of this relation_type referring to this Item_version as 'relating' and at most one Item_version_relationship of this relation_type referring as 'related';
  - 'supplied item': The application object defines a relationship between two Item_version objects representing the same object in different organizational contexts.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Item_version_relationship.
- change : Change [0..*] The change specifies the change for which this object references a modified object and the corresponding original object.

#### *Associations*

- related : Item_version [1] The related specifies the second of the two Item_version objects related by the Item_version_relationship.

## 8.3 Package Part_structure

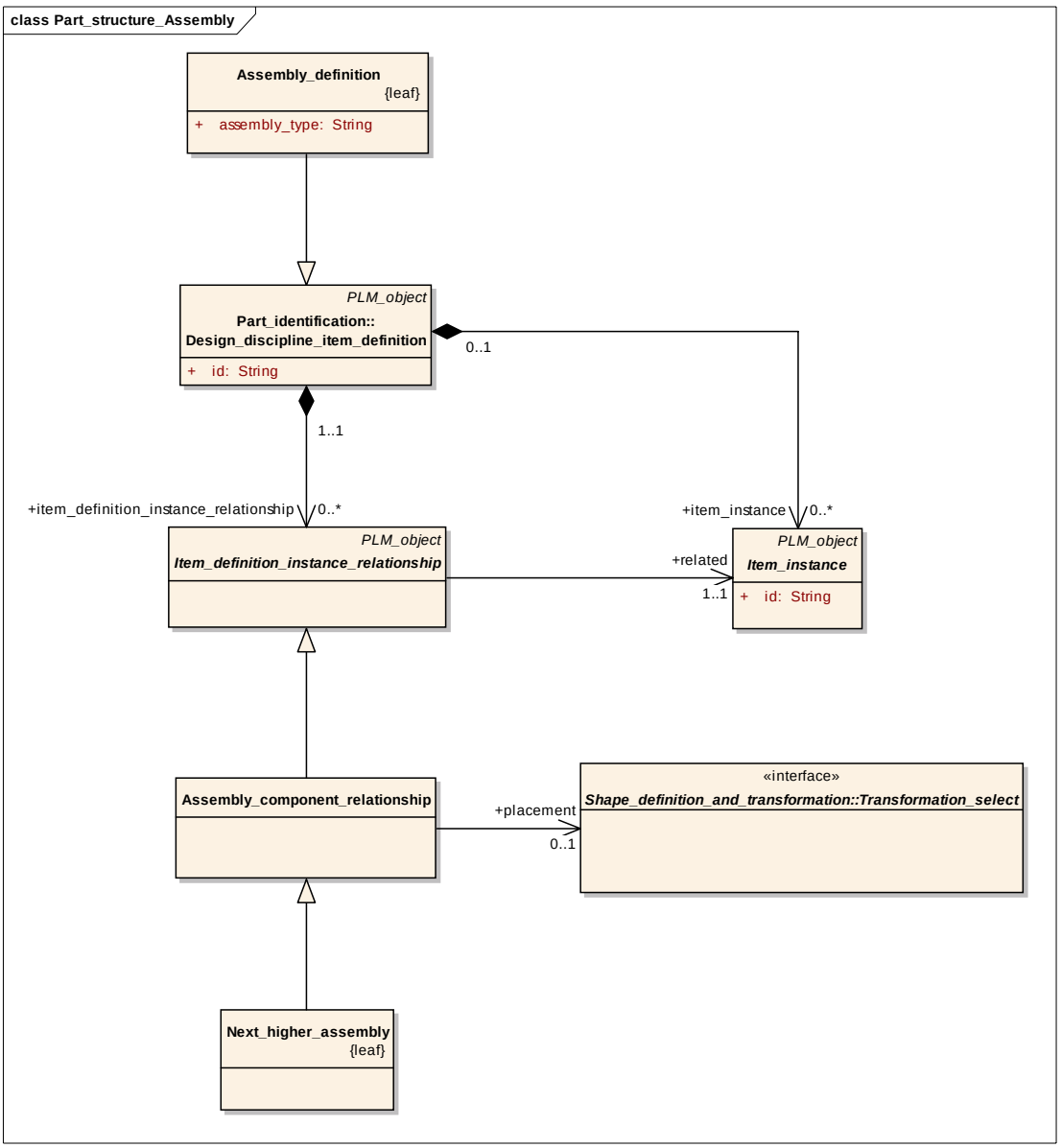


**Figure 8.18 - Part structure - Assembly**

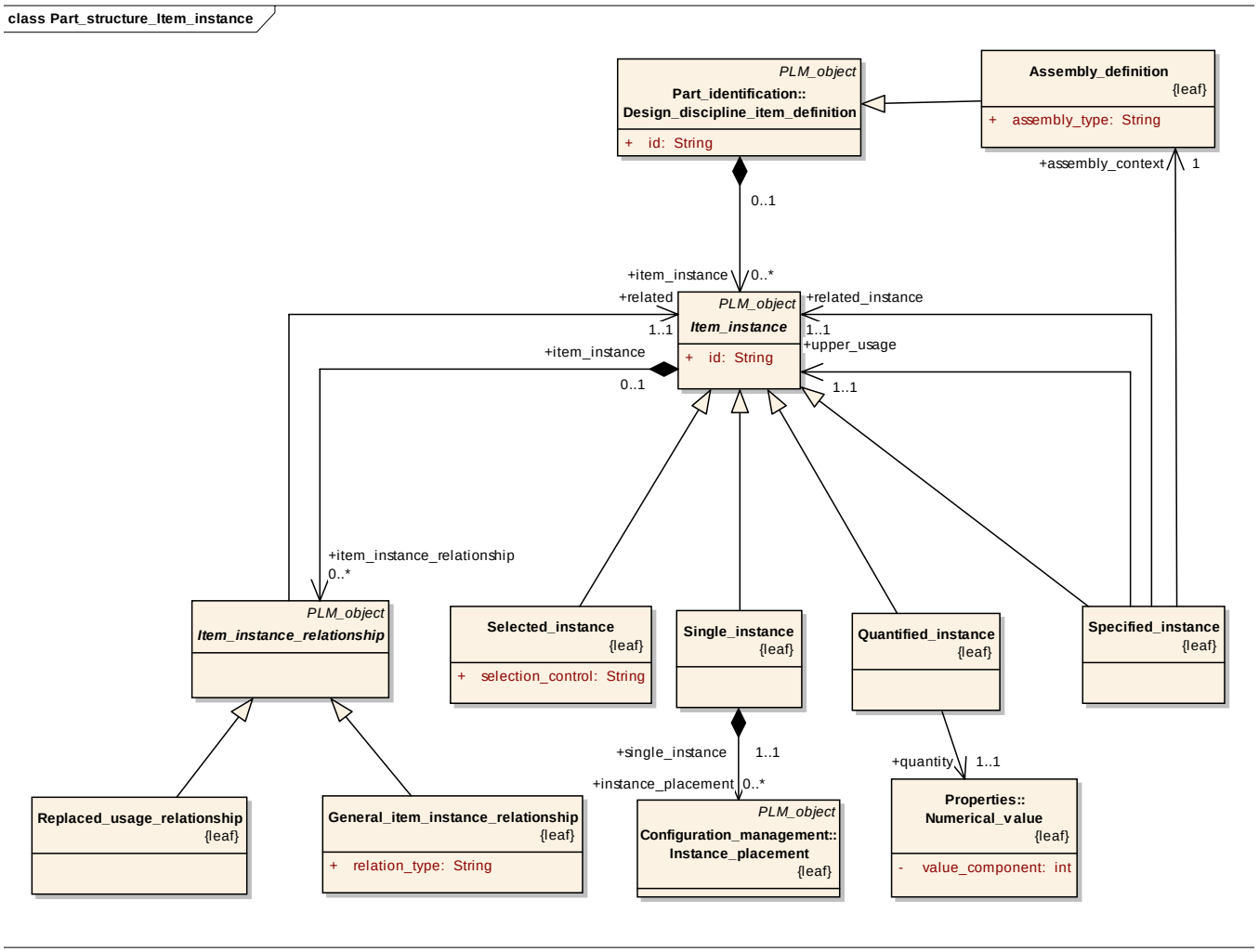


**Figure 8.19 - Part structure - Item instance**

## 8.3.1 Classes

### 8.3.1.1 Class Alternate_item_relationship

An Alternate_item_relationship is the relationship between two Item objects specifying that all versions of the alternate Item may be used in place of the base Item, regardless of its usages and of its versions.

NOTE This concept usually refers to form, fit, function, and quality. Additional properties such as performance, noise, endurance, or reliability may also be considered as a prerequisite for the replacement.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- fulfilled_requirements: string[1]: The fulfilled_requirements specifies the word or group of words describing the requirements that are covered by both the base and the alternate and therefore are the basis for the statement regarding the capability of replacement.

#### *Compositions*

#### *Associations*

- alternate: Item[1]: The alternate specifies the Item that may be used in place of the base Item

### 8.3.1.2 Class Assembly_component_relationship

An Assembly_component_relationship is the relation between an Assembly_definition and an Item_instance representing a constituent of the assembly. The Assembly_definition and the Design_discipline_item_definition that serves as 'definition' of the Item_instance shall share at least one Application_context.

#### *Base Class*

- Item_definition_instance_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- placement : Transformation_select [0..1]
  The placement specifies the Geometric_model_relationship_with_transformation or the Template_instance that specify the transformation information which is used to locate and orient the constituent in the coordinate space of the Assembly_definition. In the case of a Template_instance, the scale factor shall be omitted or set to 1.0.

### 8.3.1.3 Class Assembly_definition

An Assembly_definition is a definition of an Item_version that contains other subordinate objects.

#### *Base Class*

- Design_discipline_item_definition

#### *Attributes*

- assembly_type : String [0..1] The assembly_type specifies the kind of the Assembly_definition.

#### *Compositions*

- none

#### *Associations*

- none

### 8.3.1.4 Class Assembly_substitute_relationship

An Assembly_substitute_relationship is a relationship that indicates that one Assembly_component_relationship may be substituted for another Assembly_component_relationship. The subjects of the substitution are the related Item_instance objects of both Assembly_component_relationship objects. The relating Assembly_definition shall be the same in both Assembly_component_relationship objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

#### *Compositions*

- description: String_selectString_select[0..1]: The description specifies additional information about the Assembly_substitute_relationship.

#### *Associations*

- substitute: Assembly_component_relatiounship[1]: The substitute specifies the Assembly_component_relationship that may be used instead of the base Assembly_component_relationship.

### 8.3.1.5 Class Collected_item_association

A Collected_item_association is a mechanism to associate Item_instance objects with a Collection_definition.

#### *Base Class*

- Item_definition_instance_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.3.1.6 Class Collection_definition

A Collection_definition is the definition of an Item_version that serves as a collector for Item_instance objects that are mounted in the same vehicle but may not be assembled together.

#### *Base Class*

- Design_discipline_item_definition

#### *Attributes*

- none

#### *Compositions*

- purpose : String_select [0..1] The purpose specifies the rationale behind the Collection_definition.

#### *Associations*

- none

### 8.3.1.7 Class General_item_definition_instance_relationship

A General_item_definition_instance_relationship is a relationship between a Design_discipline_item_definition and an Item_instance whose meaning is defined by the attribute 'relation_type'.

#### *Base Class*

- Item_definition_instance_relationship (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the General_item_definition_instance_relationship.

#### *Associations*

- none

### 8.3.1.8 Class General_item_definition_relationship

A General_item_definition_relationship is a relationship between two Design_discipline_item_definition objects whose meaning is defined by the attribute 'relation_type'.

#### *Base Class*

- Item_definition_relationship (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship.

#### *Compositions*

- description : String_select [0..1]
  The description specifies additional information about the General_item_definition_relationship.

#### *Associations*

- none

### 8.3.1.9 Class General_item_instance_relationship

A General_item_instance_relationship is a relationship between two Item_instance objects whose meaning is defined by the attribute 'relation_type'.

#### *Base Class*

- Item_instance_relationship (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the General_item_instance_relationship.

#### *Associations*

- none

### 8.3.1.10 Class Item_definition_instance_relationship (ABS)

An Item_definition_instance_relationship is a relationship between a Design_discipline_item_definition and an Item_instance.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Item_definition_instance_relationship.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- related : Item_instance (ABS) [1]
  The related specifies the Item_instance that is part of the Item_definition_instance_relationship.

#### 8.3.1.11 Class Item_instance (ABS)

An Item_instance is the occurrence of an object in a product structure that is defined either by a Design_discipline_item_definition or by a Product_identification.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- id : String [1] The id specifies the identifier of the Item_instance.

##### *Compositions*

- item_instance_relationship : Item_instance_relationship (ABS) [0..*]
  The item_instance_relationship specifies the item_instance_relationship that relates the first of the two Item_instance objects.
- description : String_select [0..1]
  The description specifies additional information about the Item_instance.
- manufacturing_configuration : Manufacturing_configuration (ABS) [0..*]
  The Manufacturing_configuration specifies the Manufacturing_configuration that controls this Item_instance.
- configuration : Configuration [0..*]
  The configuration specifies the configuration that controls this Item_instance for its valid usage.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Item_instance.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Item_instance.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

##### *Associations*

- none

#### 8.3.1.12 Class Item_instance_relationship (ABS)

An Item_instance_relationship is a relationship between two Item_instance objects.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- none

#### *Compositions*

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Item_instance_relationship.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- related : Item_instance (ABS) [1]
  The related specifies the second of the two objects related by the Item_instance_relationship.

### 8.3.1.13 Class Make_from_relationship

A Make_from_relationship is a relationship between a Design_discipline_item_definition which provides the definition of a raw material, or of a semi-finished item and a Design_discipline_item_definition which provides the definition of an object manufactured out of that material, or semi-finished item.

#### *Base Class*

- Item_definition_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Make_from_relationship.

#### *Associations*

- none

### 8.3.1.14 Class Next_higher_assembly

A Next_higher_assembly is a relationship where the attribute 'related' specifies a constituent of an assembly and the attribute 'relating' specifies the immediate parent assembly of the constituent.

#### *Base Class*

- Assembly_component_relationship

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.3.1.15 Class Physical_assembly_relationship

A Physical_assembly_relationship is a mechanism to relate one Physical_instance as a component to another Physical_instance that plays the role of an assembly.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Physical_assembly_relationship.

#### *Associations*

- physical_component : Physical_instance [1]
  The physical_component specifies the Physical_instance that serves as a component in the physical structure.
- is_realization_of : Item_instance (ABS) [1]
  The is_realization_of specifies the Item_instance the physical component is the realization of.

### 8.3.1.16 Class Promissory_usage

A Promissory_usage is a relationship between a constituent of an assembly and the assembly itself. The relationship describes the intention to use the constituent in an assembly.

NOTE There may not be any geometric information provided for the constituent or the assembly that are associated by a Promissory_usage.

EXAMPLE In early design stages, the Promissory_usage may be used to create preliminary BOMs for prototyping.

A Promissory_usage is a type of Assembly_component_relationship.

#### *Base Class*

- Assembly_component_relationship

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.3.1.17 Class Quantified_instance

- A Quantified_instance is the identification of the quantified occurrence of an object that is defined either as a Design_discipline_item_definition or as a Product_identification.

#### *Base Class*

- Item_instance (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- quantity : Numerical_value [1] The quantity specifies a Numerical_value specifying the quantity of occurrences.

### 8.3.1.18 Class Replaced_definition_relationship

A Replaced_definition_relationship is a relationship between two Design_discipline_item_definition objects where the relating Design_discipline_item_definition is replaced by the related Design_discipline_item_definition.

#### *Base Class*

- Item_definition_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- change : Change [0..*] The change specifies the change for which this object references a modified object and the corresponding original object.
- description : String_select [0..1]The description specifies additional information about the Replaced_definition_relationship.

#### *Associations*

- none

### 8.3.1.19 Class Replaced_usage_relationship

A Replaced_usage_relationship is a relationship between two Item_instance objects where the relating Item_instance is replaced by the related Item_instance.

#### *Base Class*

- Item_instance_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Replaced_usage_relationship.

#### *Associations*

- usage_context : Instance_usage_context_select [1]
  The usage_context specifies the object that identifies the context in which the replacement is applicable. In the case, where the usage_context refers to a Process_operation_input_or_output, the 'relating' Item_instance shall be referred to as 'element' by the Process_operation_input_or_output. In the case, where the usage_context refers to an Item_definition_instance_relationship, the 'relating' Item_instance shall be referred to as 'related' by the Item_definition_instance_relationship. In the case, where the usage_context refers to a Product_structure_relationship, the 'relating' Item_instance shall be referred to as 'related' by the Product_structure_relationship.

### 8.3.1.20 Class Selected_instance

A Selected_instance is the identification of the occurrence of an object that is either defined as a Design_discipline_item_definition or as a Product_identification and whose quantity depends on certain constraints.

#### *Base Class*

- Item_instance (ABS)

#### *Attributes*

- selection_control : String [1] The selection_control specifies the constraint that has to be evaluated for the Selected_instance.

#### *Compositions*

- none

#### *Associations*

- selected_quantity : Value_with_unit (ABS) [1]
  The selected_quantity specifies the quantity of the part, tool or raw material foreseen as Selected_instance. The selected_quantity shall be of type Value_limit or Value_range.

### 8.3.1.21 Class Single_instance

A Single_instance is one particular occurrence of an object that is defined either as a Design_discipline_item_definition or as a Product_identification.

#### *Base Class*

- Item_instance (ABS)

#### *Attributes*

- none

#### *Compositions*

- instance_placement : Instance_placement [0..*]
  The instance_placement specifies the instance_placement which this Single_instance is placed with.

#### *Associations*

- none

### 8.3.1.22 Class Specified_instance

A Specified_instance is a mechanism to identify a certain Item_instance in a multi level assembly structure that reuses partial decompositions.

#### *Base Class*

- Item_instance (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- assembly_context : Assembly_definition [1]
  The assembly_context specifies an Assembly_definition object in which the instance identified by this mechanism is used.
- related_instance : Item_instance (ABS) [1]
  The related_instance specifies the Item_instance that is to be identified.
- upper_usage : Item_instance (ABS) [1]
  The upper_usage specifies the Item_instance in which the related_instance is used. This Item_instance shall be the immediate upper level instance or another Specified_instance.

### 8.3.1.23 Class Tool_part_relationship

A Tool_part_relationship is a relationship between two Design_discipline_item_definition objects. It establishes a relationship between an item (related) and a tool (relating) that is used to produce the item.

#### *Base Class*

- Item_definition_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- used_technology_description : String_select [0..1]
  The used_technology_description specifies the technology that is used to manufacture the part using this tool and, possibly, the reasons for the use of a particular technology.

#### *Associations*

- placement : Transformation_target_select [0..1]
  The placement specifies the relative position of the Item representing the part with respect to the local coordinate system of the Item representing the tool.

### 8.3.2 Interfaces

#### 8.3.2.1 Interface Instance_usage_context_select

This empty interface is realized by the following classes:

- Product_structure_relationship
- Item_definition_instance_relationship (ABS)
- Process_operation_input_or_output

#### 8.3.2.2 Interface Item_information_select

This empty interface is realized by the following classes:

- Product_component
- Physical_instance
- Design_discipline_item_definition
- Item_instance (ABS)

#### 8.3.2.3 Interface Product_constituent_select

This empty interface is realized by the following classes:

- Product_function
- Product_component
- Item_instance (ABS)

## 8.4 Package Document_and_file_management

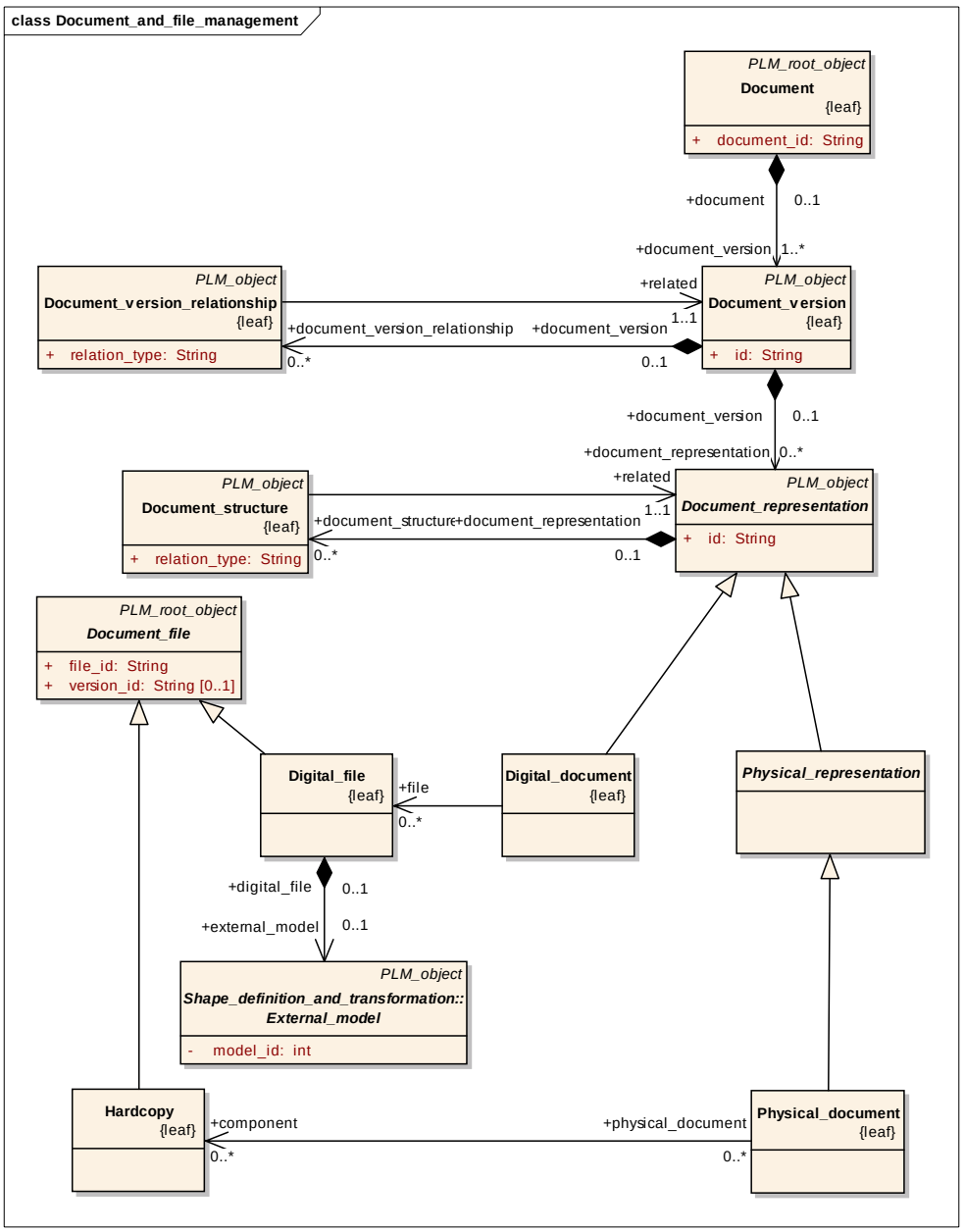


**Figure 8.20 - Document and file management**

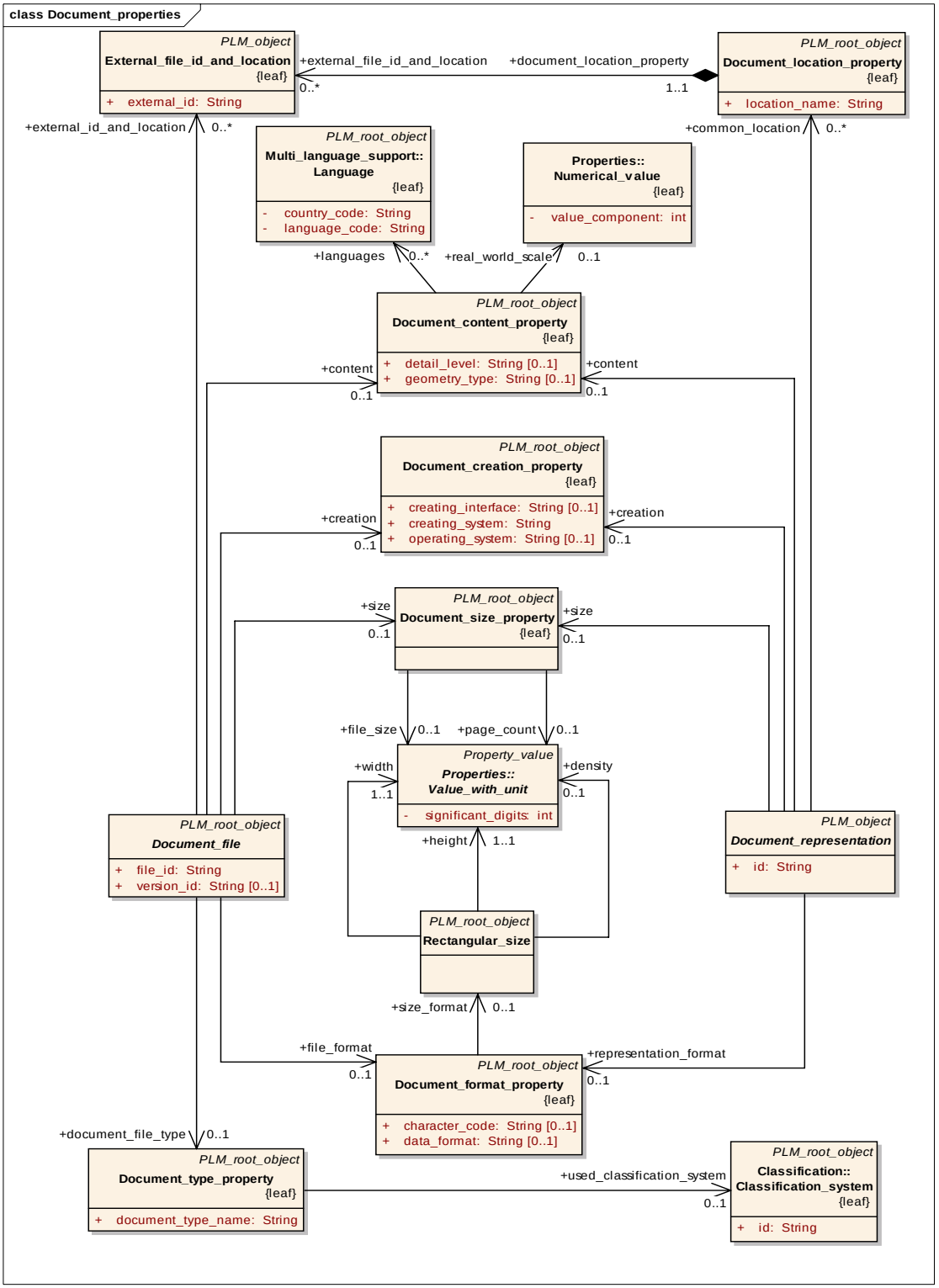


**Figure 8.21 - Document properties**

### 8.4.1 Classes

#### 8.4.1.1 Class Digital_document

A Digital_document is a piece of product data that is archived in a digital format.

***Base Class***

- Document_representation (ABS)

***Attributes***

- none

***Compositions***

- none

***Associations***

- file : Digital_file [0..*] The file specifies a computer interpretable realization of the Digital_document.

#### 8.4.1.2 Class Digital_file

A Digital_file contains computer interpretable data.

***Base Class***

- Document_file (ABS)

***Attributes***

- none

***Compositions***

- external_model : External_model (ABS) [0..1]
  The external_model specifies the externally defined geometry information contained in this Digital_file.

***Associations***

- none

#### 8.4.1.3 Class Document

A Document is a logical document that serves as the identifier for a container for some product data.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- document_id : String [1] The document_id specifies the identifier of the Document.

#### *Compositions*

- document_version : Document_version [1..*]
  The document_version specifies the document_version of this logical document.
- name : String_select [1] The name specifies the word or group of words by which the Document is referred to.
- description : String_select [0..1]The description specifies additional information about the Document.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Document.

#### *Associations*

- none

### 8.4.1.4 Class Document_assignment

A Document_assignment is a mechanism to associate a document with an object, where the assigned document provides information about the object it is associated to.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1] The role specifies the meaning of the Document_assignment:
  - 'additional information': The assigned document provides information that is relevant for the associated object, but is not a description of the associated object itself;
  - 'behavior': The assigned document specifies information about the behavior of the associated object;
  - 'description': The assigned document provides textual information for the associated object it-self;
  - 'informative': The assigned document may or may not be considered;
  - 'mandatory': The associated object shall conform to the content of the assigned document;
  - 'mathematical description': The assigned document specifies the associated object by providing the algorithmic specification of its behavior

#### *Compositions*

- none

#### *Associations*

- assigned_document : Assigned_document_select [1]
  The assigned_document specifies the Document, a Document_version, a Document_representation, or a Document_file that is used to provide information.
- is_assigned_to : Documented_element_select [1..*]
  The documented elements.

#### 8.4.1.5 Class Document_content_property

A Document_content_property specifies characteristics precising the content of a Document_file or of a Document_representation. At least one of the optional attributes shall be specified for each instance of this object.

##### *Base Class*

- PLM_root_object (ABS)

##### *Attributes*

- detail_level : String [0..1] The detail_level specifies the level of detail that the Document_file or the Document_representation provides. Where applicable the following values shall be used:
  - 'rough 3d shape': 3D shape model without edge rounds and fillets;
  - 'rounded edges': 3D shape model with edge rounds and fillets.
- geometry_type : String [0..1] The geometry_type specifies the kind or kinds of geometry that an object contains. Where applicable the following values shall be used:
  - '3D wireframe model': The document contains a 3D shape model in wireframe representation;
  - '2D shape': The document contains a 2D shape model or contours only;
  - 'surface model': The document contains a 3D shape model in surface representation;
  - 'closed volume': The document contains a 3D shape model in closed body topological surface representation;
  - 'solid model': The document contains a 3D shape model in advanced boundary representation;
  - 'solid and surface model': The document contains a 3D shape model in surface and advanced boundary representation;
  - 'assembly': The document contains an assembly structure with reference to the assembled components and their transformation matrices;
  - 'assembly with mating elements': The document contains an assembly structure including the mating components only, such as screws or rivets, with exact positioning information. This assembly representation is intended to be overlayed with the assembly structure for the main components;
  - '2D drawing': The document contains a technical drawing without 3D shape representation;
  - 'drawing derived from 3D data': The document contains a technical drawing that has been derived from a 3D shape model;
  - 'drawing related to 3D data': The document contains a technical drawing that visualizes a 3D shape model and possibly establishes associative links to the 3D shape model.

##### *Compositions*

- none

##### *Associations*

- languages : Language [0..*] The languages specifies which language or languages are used in the characterized objects.
- real_world_scale : Numerical_value [0..1]The real_world_scale specifies the scale that is used in the Document_file or in the Document_representation the Document_content_property is referred by.

#### 8.4.1.6 Class Document_creation_property

A Document_creation_property specifies characteristics of Document_file or of Document_representation objects. It specifies the context of the creation of the object. At least one of the optional attributes shall be specified for each instance of this object.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- creating_system : String [1] The creating_system specifies the computer application or the machine which is used to create the object that is characterized.
- operating_system : String [0..1]The operating_system specifies the operating system that is used to execute the computer application that created the characterized object.
- creating_interface : String [0..1]The creating_interface specifies the computer application used to create the Document_file or Document_representation object.

***Compositions***

- none

***Associations***

- none

#### 8.4.1.7 Class Document_file (ABS)

A Document_file is one of potentially more files on a computer system or in actual stacks of paper that make up a Document_representation.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- file_id : String [1] The file_id specifies the identifier which is used to locate the file either on a computer system or in a repository of paper documents.
- version_id : String [0..1] The version_id specifies the identification of the version that distinguishes one Document_file object from other versions of Document_file objects with the same file_id.

***Compositions***

- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

***Associations***

- creation : Document_creation_property [0..1]
  The creation specifies further details of the context of the creation of the Document_file.

- content : Document_content_property [0..1]
  The content characterizes the content of the Document_file.
- file_format : Document_format_property [0..1]
  The file_format specifies the characteristics of the Document_file that specify the format of the object.
- size : Document_size_property [0..1]
  The size specifies characteristics for the size of the Document_file.
- external_id_and_location : External_file_id_and_location [0..*]
  The external_id_and_location specifies alternatives of the identifier and location of the Document_file.
- document_file_type : Document_type_property [0..1]
  The document_file_type specifies the format of the Document_file. It shall only be specified, if the Document_file does not participate in a Document.

#### 8.4.1.8 Class Document_file_relationship

A Document_file_relationship is a relationship between two Document_file objects. It specifies that the related Document_file is referenced from the relating Document_file.

EXAMPLE A service manual may contain graphics for explanatory reasons. In this case the Document_file objects that contain the graphics are referenced as related from the Document_file object that contains the body of the service manual with relation_type 'reference'.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- relation_type: string[1]: The relation_type specifies the meaning of the relationship. The values 'addition', 'decomposition', and 'peer' are only allowed if neither the relating nor the related Document_file are included in a Document_representation.

##### *Compositions*

- description: String_select[0..1]: The description specifies additional information about the Document_file_relationship.

##### *Associations*

- related : Document_file[1] The related specifies the second of the two objects related by the Document_file_relationship.

#### 8.4.1.9 Class Document_format_property

A Document_format_property specifies characteristics of a Document_file or of a Document_representation that specify the format of the object. At least one of the optional attributes shall be specified for each instance of this object.

##### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- data_format : String [0..1] The data_format specifies the convention that was used to structure the information in the characterized object. Where applicable the following values shall be used:
  - 'DXF': The document contains data in Drawing Exchange File format;
  - 'IGES': The document contains data in Initial Graphics Exchange Specification format;
  - 'ISO 10303-203': The document contains data in ISO 10303-203 format;
  - 'ISO 10303-214': The document contains data in ISO 10303-214 format;
  - 'TIFF CCITT GR4': The document contains data in TIFF CCITT GR4 format;
  - 'VDAFS': The document contains data in VDAFS format;
  - 'VOXEL': The document contains data in VOXEL format.
- character_code : String [0..1] The character_code specifies the character code that is used in the characterized object. Where applicable the following values shall be used:
  - 'binary': The document contains data in binary format;
  - 'IEC 61286': The coded character set used to encode the document data according to IEC 61286;
  - 'ISO 646': The coded character set used to encode the document data according to ISO 646;
  - 'ISO 3098-1': The coded character set used to encode the document data is according to ISO 3098-1;
  - 'ISO 6937': The coded character set used to encode the document data is according to ISO/IEC 6937;
  - 'ISO 8859-1': The coded character set used to encode the document data according to ISO 8859-1;
  - 'ISO 10646': The coded character set used to encode the document data according to ISO/IEC 10646.

#### *Compositions*

- none

#### *Associations*

- size_format : Rectangular_size [0..1]
  The size_format specifies the dimensions of a physical presentation of the object the size_format is provided for.

### 8.4.1.10 Class Document_location_property

A Document_location_property specifies where a Document_file or a Document_representation can be found in a digital or physical data storage system.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- location_name : String [1] The location_name specifies the location, where the object that refers to the Document_location_property, can be found. 'C:\mpbs{}programs' and '/usr/local/bin' are examples for a location_name.

#### *Compositions*

- external_file_id_and_location : External_file_id_and_location [0..*]
  The external_file_id_and_location specifies the Document_file that is stored in this Document_location_property.

#### *Associations*

- none

### 8.4.1.11 Class Document_representation (ABS)

A Document_representation is one of potentially more alternative representations of a Document_version.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Document_representation.

#### *Compositions*

- document_structure : Document_structure [0..*]
  The document_structure specifies the document_structure that relates the first of the two Document_representation objects.
- description : String_select [0..1]The description specifies additional information about the Document_representation.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Document_representation.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- content : Document_content_property [0..1]
  The content specifies characteristics of the content of the Document_representation.
- size : Document_size_property [0..1]
  The size specifies the size of the represented document.
- representation_format : Document_format_property [0..1]
  The representation_format specifies the format of the document represented by Document_representation.
- common_location : Document_location_property [0..*]
  The common_location specifies the location of a Document_representation, where all its constituents can be found.
- creation : Document_creation_property [0..1]
  The creation specifies further details of the creation of the Document_representation.

#### 8.4.1.12 Class Document_size_property

A Document_size_property specifies the size of a Document_file or of a Document_representation object. At least one of the optional attributes shall be specified for each instance of this object.

##### *Base Class*

- PLM_root_object (ABS)

##### *Attributes*

- none

##### *Compositions*

- none

##### *Associations*

- page_count : Value_with_unit (ABS) [0..1]
  The page_count specifies the number of pages of the application object the Document_size_property is referred by. The page_count shall only be used in cases where the Document_size_property is referred by a Hardcopy or a Physical_representation.
- file_size : Value_with_unit (ABS) [0..1]
  The file_size specifies the Value_with_unit that represents the size of a digitally stored document. The file_size shall only be applied in cases where the Document_size_property is referred by a Digital_document or a Document_file.

#### 8.4.1.13 Class Document_structure

A Document_structure is a relationship between two Document_representation objects.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- relation_type : String [1]
  The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'addition': The application object specifies that the related document provides supplementary or collateral information with regard to the information provided by the relating document;
  - 'copy': The application object defines a relationship where the related Document_representation is a copy of the relating Document_representation;
  - 'decomposition': The application object defines a relationship where the related Document_representation is one of potentially more sub documents of the relating Document_representation;
  - 'derivation': The application object defines a relationship where the related Document_representation is derived from the relating Document_representation;
  - 'peer': The application object specifies that the related document provides required information with regard to that provided by the relating document. The peer document is essential for a complete understanding;

- 'reference': The application object defines a relationship where the related document is referenced from the relating;
- 'sequence': The application object defines a logical sequence where the related Document_representation comes after the relating Document_representation;
- 'substitution': The application object defines a relationship where the related Document_representation replaces the relating Document_representation;
- 'translation': The Document_structure specifies that the related document is generated through a translation process from the relating document.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Document_structure.

#### *Associations*

- related : Document_representation (ABS) [1]
  The related specifies the second of the two objects related by the Document_structure.

### 8.4.1.14 Class Document_type_property

A Document_type_property specifies the kind of a Document_file.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- document_type_name : String [1]
  The document_type_name specifies the word or the group of words that describe the kind of object the characteristics are provided for. Where applicable the following values shall be used:
  - 'geometry': The document represents a shape model;
  - 'NC data': The document represents numerical control data;
  - 'FE data': The document represents finite element data;
  - 'sample data': The document represents measured data;
  - 'process plan': The document represents process planning data;
  - 'check plan': The document represents quality control planning data;
  - 'drawing': The document represents a technical drawing.

#### *Compositions*

- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Document_type_property.

#### *Associations*

- used_classification_system : Classification_system [0..1]
  The used_classification_system specifies the Classification_system the document_type_name is defined in.

#### 8.4.1.15 Class Document_version

A Document_version is a release of a Document.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- id : String [1] The id specifies the identifier of the Document_version. The id shall be unique within the scope of the associated Document.

##### *Compositions*

- document_version_relationship : Document_version_relationship [0..*]
  The document_version_relationship specifies the document_version_relationship that relates the first of the two Document_version objects.
- description : String_select [0..1]The description specifies additional information about the Document_version.
- document_representation : Document_representation (ABS) [0..*]
  The document_representation specifies the document_representation that represents this version of the logical document.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Document_version.

##### *Associations*

- none

#### 8.4.1.16 Class Document_version_relationship

A Document_version_relationship is a relationship between two Document_version objects.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'derivation': The application object defines a deriving relationship where the related Document_version is based on the relating Document_version which is an earlier version of the same or of a different Document;
  - 'hierarchy': The application object defines a hierarchical relationship where the related Document_version is a sub version of the relating Document_version;
  - 'sequence': The application object defines a version sequence where the relating Document_version is the preceding version and the related Document_version is the following version.;
  - 'supplied document': The application object defines a relationship between two

Document_version objects representing the same object in different organizational contexts.

#### *Compositions*

- description : String_select [0..1] The description specifies additional information about the Document_version_relationship.

#### *Associations*

- related : Document_version [1] The related specifies the second of the two objects related by the Document_version_relationship.

### 8.4.1.17 Class External_file_id_and_location

An External_file_id_and_location specifies the location of a file in an external storage system.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- external_id : String [0..1] The external_id specifies the identifier of a document in an external storage system.

#### *Compositions*

- none

#### *Associations*

- none

### 8.4.1.18 Class Hardcopy

A Hardcopy is the actual stack of paper consisting of one or more sheets, on which some product data is written, printed or plotted.

#### *Base Class*

- Document_file (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

#### 8.4.1.19 Class Named_size

A Named_size is the definition of the size of a Document_file or of a Document_representation where the size is specified by a standardized identifier.

##### *Base Class*

- Rectangular_size

##### *Attributes*

- size : String [1] The size specifies the size of the object. If the size differs from the dimensions specified by the inherited 'width' and 'height' attributes the size is overridden.

##### *Compositions*

- none

##### *Associations*

- referenced_standard : Classification_system [0..1]
  The referenced_standard specifies a standard according to which the size is specified.

#### 8.4.1.20 Class Partial_document_assignment

A Partial_document_assignment identifies a specific portion of the contents of a document. It assigns this portion to the product data for which it is relevant.

A Partial_document_assignment is a type of Document_assignment.

##### *Base Class*

- Document_assignment

##### *Attributes*

- document_portion: string[1]: The document_portion specifies the word or group of words that convey the subject or sub contents of the Document. EXAMPLE If the assigned document is a service manual, the document_portion may specify that, in the given situation, only the information related to electric fuses is relevant among the overall contents of the document.

##### *Compositions*

- none

##### *Associations*

- none

#### 8.4.1.21 Class Physical_document

A Physical_document is a piece of product data that is archived in a non-digital form.

#### *Base Class*

- Physical_representation (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- component : Hardcopy [0..*]  The component specifies the physical realization of the Physical_document.

### 8.4.1.22 Class Physical_model

A Physical_model is a model of the shape of a part made from a certain material, e.g., clay or wood, at a certain scale. For a Physical_model, neither a Document_size_property, a Document_format_property, nor a Document_creation_property shall be specified.

EXAMPLE A Physical_model may be used to get an impression of the future appearance of, e.g., a car or to use it, e.g., in a wind-tunnel.

A Physical_model is a type of Physical_representation.

#### *Base Class:*

- Physical_representation (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.4.1.23 Class Physical_representation (ABS)

A Physical_representation is a physically realizable representation of a Document_version.

#### *Base Class*

- Document_representation (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

#### 8.4.1.24 Class Rectangular_size

A Rectangular_size is the definition of the planar size of an object.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- density : Value_with_unit (ABS) [0..1]
  The density specifies the resolution of the object if it is a raster picture.
- height : Value_with_unit (ABS) [1]
  The height specifies the size of the object in vertical direction.
- width : Value_with_unit (ABS) [1]The width specifies the size of the object in horizontal direction.

### 8.4.2 Interfaces

#### 8.4.2.1 Interface Assigned_document_select

This empty interface is realized by the following classes:

- Document_version
- Document_representation (ABS)
- Document_file (ABS)
- Document

#### 8.4.2.2 Interface Documented_element_select

#### *Compositions*

- document_assignment : Document_assignment [0..*]

#### *Implemented By*

- Shape_element_relationship

- Process_operation_occurrence
- Work_order
- Product_identification
- Organization
- Contract
- Physical_instance_test_result
- Item_definition_instance_relationship
- Complex_product
- Classification_attribute
- Item
- Product_class
- Class_structure_relationship
- Item_definition_relationship
- Specification_category
- Change
- Specific_item_classification
- Material
- Specification
- Item_version
- Activity_element
- Project
- Classification_system
- Security_classification
- Process_plan
- Activity_method
- Approval
- Item_instance
- Descriptive_specification
- Property
- Product_structure_relationship
- Shape_element
- Shape_element_relationship
- General_classification
- Design_discipline_item_definition

- Item_instance_relationship
- Physical_instance
- Work_request
- Certification
- Item_shape
- Design_constraint
- Physical_assembly_relationship
- Activity
- Retention_period
- Class_structure_relationship
- Person

## 8.5 Package Shape_definition_and_transformation

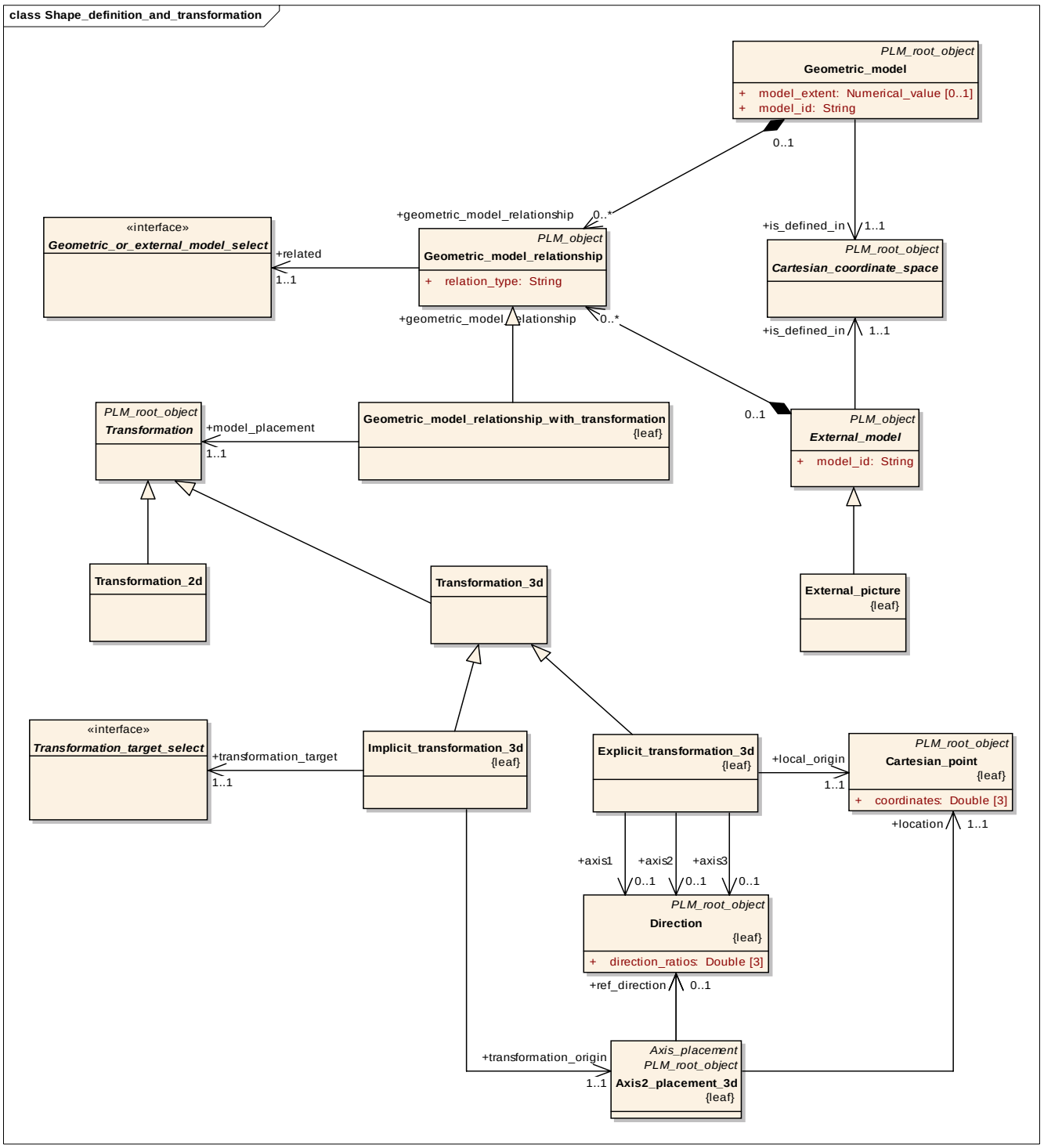


**Figure 8.22 - Shape definition and transformation**

## 8.5.1 Classes

### 8.5.1.1 Class Accuracy

An Accuracy is the information about the geometrical accuracy of the product data contained in a model.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- accuracy_value : Double [1] The accuracy_value specifies a numerical value defining the Accuracy.
- accuracy_type : String [1] The accuracy_type specifies the kind of accuracy that is applied. Where applicable the following values shall be used:
  - 'angular accuracy': A kind of accuracy that specifies the maximum value for the absolute angle between two curve tangents or two surface normals for which the creating system assumes curve tangents or surface normals being identical;
  - 'curvature accuracy': A kind of accuracy that specifies the value for the term under which a system can assume that the two radii of curvature R1 and R2 are identical. The curvature accuracy value is used to determine the accuracy range for curvature continuous curve or surface connections;
  - 'distance accuracy': A kind of accuracy that specifies the distance under which two points can be considered as having the same location. The distance accuracy value defined for a Geometric_model is valid for all geometric elements of the Geometric_model.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Accuracy.

#### *Associations*

- is_defined_for : Accuracy_select [1..*]
  The is_defined_for specifies the geometry to which the Accuracy is assigned.

### 8.5.1.2 Class Axis_placement

An Axis_placement is the representation of three orthogonal axes defined in a three-dimensional reference coordinate system, which intersect at a unique point. This point defines the origin of the Axis_placement.

#### *Base Class:*

- PLM_root_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- location: Cartesian_point[0..1] The location specifies the geometric position of a reference point, such as the centre of a circle, of the item to be located.

### 8.5.1.3 Class Axis1_placement

An Axis1_placement represents the direction and location in three-dimensional space of a single axis. An axis1_placement is defined in terms of a locating point (inherited from the Axis_placement supertype) and an axis direction; this is either the direction of axis or defaults to (0.0,0.0,1.0). The actual direction for the axis placement is given by the derived attribute z; the normalized direction of the local Z axis.

#### *Base Class*

- Axis_placement (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- axis: Direction[0..1]: The axis specifies the direction of the local Z axis.

### 8.5.1.4 Class Axis2_placement_2d

An Axis2_placement_2D is the location and orientation in two-dimensional space of two mutually perpendicular axes. An axis2_placement_2d is defined in terms of a point, (inherited from the placement supertype), and an axis. It can be used to locate and orientate an object in two-dimensional space and to define a placement coordinate system. The entity includes a point which forms the origin of the placement coordinate system. A direction vector is required to complete the definition of the placement coordinate system. The ref_direction defines the placement X axis direction; the placement Y axis direction is derived from this.

#### *Base Class*

- Axis_placement (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations:*

- ref_direction: Direction[0..1]: The ref_direction specifies the direction used to determine the direction of the local X axis. If ref_direction is omitted, this direction is taken from the geometric coordinate system.

#### 8.5.1.5 Class Axis2_placement_3d

An Axis2_placement_3d is the location and orientation in three-dimensional space of two mutually perpendicular axes. An axis2_placement_3d is defined in terms of a point, (inherited from the placement supertype), and two (ideally orthogonal) axes. It can be used to locate and orientate a non axis-symmetric object in space and to define a placement coordinate system. The entity includes a point which forms the origin of the placement coordinate system. Two direction vectors are required to complete the definition of the placement coordinate system. The axis is the placement Z axis direction and the ref_direction is an approximation to the placement X axis direction.

NOTE Let z be the placement Z axis direction and a be the approximate placement X axis direction. There are two methods, mathematically identical but numerically different, for calculating the placement X and Y axis directions.

The vector a is projected onto the plane defined by the origin point P and the vector z to give the placement X axis direction as x = (a -(a.z)z). The placement Y axis direction is then given by y = (z x x).

The placement Y axis direction is calculated as y = (z x a) and then the placement X axis direction is given by x = (y x z).

The first method is likely to be the more numerically stable of the two, and is used here.

A placement coordinate system referenced by the parametric equations is derived from the axis2_placement_ 3d data for conic curves and elementary surfaces.

##### *Base Class*

- Axis_placement (ABS)

##### *Attributes*

- none

##### *Compositions*

- none

##### *Associations*

- ref_direction : Direction [0..1] The ref_direction can be used to determine the direction of the local X axis.
- axis : Direction [0..1] The axis defines the exact direction of the local Z axis.

#### 8.5.1.6 Class Cartesian_coordinate_space (ABS)

Cartesian_coordinate_space is a coordinate space in which geometric and annotation elements may be defined. It is either two-dimensional or three-dimensional. An origin for coordinate values is implicitly defined. The units applicable to the coordinate values of elements defined in the Cartesian_coordinate_space are specified.

##### *Base Class*

- PLM_root_object (ABS)

##### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- unit_of_values : Unit [0..*] The unit_of_values specifies the various units in which any values are expressed. The same length unit is applied to each coordinate direction. Only one unit of a kind shall be specified. In the case where geometric elements are defined in the Cartesian_coordinate_space, there shall be at least two units specified, the length unit and the plane angle unit.

### 8.5.1.7 Class Cartesian_coordinate_space_2d

A Cartesian_coordinate_space_2d is a two-dimensional coordinate space. Any two-dimensional geometric and annotation element shall be defined in a Cartesian_coordinate_space_2d.

#### *Base Class*

- Cartesian_coordinate_space (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.5.1.8 Class Cartesian_coordinate_space_3d

A Cartesian_coordinate_space_3d is a three-dimensional coordinate space. Any three-dimensional geometric data shall be defined in a Cartesian_coordinate_space_3d.

#### *Base Class*

- Cartesian_coordinate_space (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

#### 8.5.1.9 Class Cartesian_point

A Cartesian_point is a point that is defined by its coordinates in a rectangular Cartesian coordinate system.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- coordinates : Double [3] The coordinates specify the 3 coordinates of the point.

***Compositions***

- none

***Associations***

- none

#### 8.5.1.10 Class Detailed_element

A Detailed_element is a single element of a Geometric_model or of a Styled_model.

Each Detailed_element is a Detailed_model_element, a Styled_element, or a Draughting_callout.

***Base Class:***

- PLM_object (ABS)

***Attributes***

- none

***Compositions:***

- name: String_select[0..1] The name specifies the word or group of words by which the Detailed_element is referred to.

***Associations***

- none

#### 8.5.1.11 Class Detailed_geometric_model_element

A Detailed_geometric_model_element is the identification of a geometric element.

NOTE The Detailed_geometric_model_element may be used to define the elements of a Geometric_model.

A Detailed_geometric_model_element is a type of Detailed_model_element.

***Base Class***

- Detailed_model_element (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.5.1.12 Class Detailed_model_element

A Detailed_model_element is a single element of a model.

A Detailed_model_element is a type of Detailed_element.

Each Detailed_model_element is a Template_instance or a Detailed_geometric_model_element.

#### *Base Class*

- Detailed_element (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.5.1.13 Class Direction

A Direction in a 3-dimensional space is expressed as a vector.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- direction_ratios : Double [3]  The direction_ratios specify the 3 ratios of the direction vector components.

#### *Compositions*

- none

#### *Associations*

- none

#### 8.5.1.14 Class Explicit_transformation_2d

An Explicit_transformation_2d is a geometric transformation in two-dimensional space where the translation and rotation is explicitly defined.

An Explicit_transformation_2d is a type of Transformation_2d.

#### 8.5.1.15 Base Class

- Transformation_2d

##### *Attributes*

- none

##### *Compositions*

- none

##### *Associations*

- axis1 : Direction[0..1]: The axis1 specifies the X axis direction.
- axis2 : Direction[0..1]: The axis2 specifies the Y axis direction.
- local_origin : Cartesian_point[1]:
  The local_origin specifies the required translation. The actual translation includes in the transformation is from the geometric origin to the local origin.

#### 8.5.1.16 Class Explicit_transformation_3d

A geometric relationship between external models can be defined explicitly by using an Explicit_transformation_3d that has a local origin and a rotation matrix.

##### *Base Class*

- Transformation_3d

##### *Attributes*

- none

##### *Compositions*

- none

##### *Associations*

- axis3 : Direction [0..1] The axis3 is the Z axis direction of the transformation target.
- axis2 : Direction [0..1] The axis2 is the Y axis direction of the transformation target.
- axis1 : Direction [0..1] The axis1 is the X axis direction of the transformation target.
- local_origin : Cartesian_point [1]
  The local_origin is the required translation specified as a cartesian point. The actual translation included in the transformation is from the geometric origin to the local origin.

### 8.5.1.17 Class External_geometric_model

An External_geometric_model is the identification of a model that contains geometry in a 3D context only.

#### *Base Class*

- External_model (ABS)

#### *Attributes*

- model_extent : Numerical_value [0..1]
  The model_extent specifies the radius of a sphere that contains all elements of the model and whose centre is at the origin of the Cartesian_coordinate_space of the External_geometric_model. The model_extent is specified using a length unit.

#### *Compositions*

- none

#### *Associations*

- none

### 8.5.1.18 Class External_geometric_model_with_parameters

An External_geometric_model_with_parameters is the identification of a model that contains geometry in a 3D context, and where some characteristics of the model may be controlled or changed in a predefined way in each occurrence of the model.

EXAMPLE A geometric model may have predefined aspect ratios of its overall dimensions, but a parameterized volume. In each occurrence of the External_geometric_model_with_parameters, the volume may be specified by the user and the dimensions of the model are set automatically while the predefined aspect ratio requirements are met.

The usage of the controlled characteristics shall be subject of an agreement of common understanding by partners sharing this information.

An External_geometric_model_with_parameters is a type of External_geometric_model.

#### *Base Class*

- External_geometric_model

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- parameter_value: General_parameter[1]:
  The parameter_value specifies the General_parameter object or group of General_parameter objects that may be varied in each occurrence.

#### 8.5.1.19 Class External_model (ABS)

An External_model is the identification of a model that is described in a Digital_file and by the Cartesian_coordinate_space that is needed to further process the externally described information.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- model_id : String [1] The model_id specifies the identifier of the External_model.

##### *Compositions*

- geometric_model_relationship : Geometric_model_relationship [0..*]
  The geometric_model_relationship specifies the geometric_model_relationship that relates the first of the two External_model objects.
- description : String_select [0..1]The description specifies additional information about the External_model.

##### *Associations*

- is_defined_in : Cartesian_coordinate_space (ABS) [1]
  The is_defined_in specifies the Cartesian_coordinate_space that defines the context for the externally described geometry. If the External_model is an External_picture, the context shall be a Cartesian_coordinate_space_2d.

#### 8.5.1.20 Class External_picture

An External_picture is the identification of a model that is described by a two dimensional image.

##### *Base Class*

- External_model (ABS)

##### *Attributes*

- none

##### *Compositions*

- none

##### *Associations*

- none

#### 8.5.1.21 Class Geometric_model

A Geometric_model is a representation of geometry. A Geometric_model that does not reference any Detailed_geometric_model_element objects through one of the subtypes directly shall either reference at least one Template_instance as 'additional_element' or shall reference Axis_placement objects exclusively.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- model_id : String [1] The model_id specifies the identifier of the Geometric_model.
- model_extent : Numerical_value [0..1]
  The model_extent specifies the radius of a sphere that contains all elements of the model and whose centre is at the origin of the Cartesian_coordinate_space of the Geometric_model. The model_extent is specified using a length unit.

#### *Compositions*

- description : String_select [0..1]
  The description specifies additional information about the Geometric_model.
- geometric_model_relationship : Geometric_model_relationship [0..*]
  The geometric_model_relationship specifies the geometric_model_relationship that relates the first of the two Geometric_model objects.

#### *Associations*

- is_defined_in : Cartesian_coordinate_space (ABS) [1]
  The is_defined_in specifies the Cartesian_coordinate_space in which the Geometric_model is defined. The specified Cartesian_coordinate_space serves also as the reference coordinate space for the transformation of Template_instance objects used as additional elements in the Geometric_model.

### 8.5.1.22 Class Geometric_model_relationship

A Geometric_model_relationship is a relationship between two models. The models may be either of type Geometric_model or of type External_model.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship.

#### *Compositions*

- description : String_select [0..1]
  The description specifies additional information about the Geometric_model_relationship.

#### *Associations*

- related : Geometric_or_external_model_select [1]
  The related specifies the second of the two model objects related by the Geometric_model_relationship.

#### 8.5.1.23 Class Geometric_model_relationship_with_transformation

A Geometric_model_relationship_with_transformation is a relationship between two model objects with the additional information about a geometric Transformation. This Transformation defines the location and orientation of the related model relative to the relating model.

***Base Class***

- Geometric_model_relationship

***Attributes***

- none

***Compositions***

- none

***Associations***

- model_placement : Transformation (ABS) [1]
  The model_placement specifies the geometric Transformation that places and orients the related model relative to the relating model.

#### 8.5.1.24 Class Geometric_model_version

A Geometric_model_version is a particular version of a Geometric_model.

A Geometric_model_version is a type of Geometric_model.

***Base Class***

- Geometric_model

***Attributes***

- version_id: string[1]: The version_id specifies the identification of a particular version of a Geometric_model.

***Compositions***

- none

***Association***

- none

#### 8.5.1.25 Class Geometrical_relationship

A Geometrical_relationship is the relationship between two Design_discipline_item_definition objects specifying two parts that are geometrically related.

***Base Class***

- Item_definition_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Geometrical_relationship.

#### *Associations*

- definition_placement : Transformation_select [1]
  The definition_placement specifies the Geometric_model_relationship_with_transformation or the Template_instance that has the Transformation to be applied to the relating Design_discipline_item_definition in order to define the location and the orientation of the related Design_discipline_item_definition. Translation, rotation, and mirroring, i.e., inversion, is included; scaling is not included. In the case of a Template_instance, the scale factor shall be omitted or set to 1.0.

### 8.5.1.26 Class Implicit_transformation_2d

An Implicit_transformation_2d is a transformation that is defined by two instances of axis2_placement_2d in which one instance of axis2_placement_2d is the result of applying the transformation function to the other. The transformation function is not explicitly provided, but it is derived from the relationship between the instances of axis2_placement_2d. The transformation between the two instances of axis2_placement_2d applies to every element in the objects related by the Implicit_transformation_2d.

#### *Base Class*

- Transformation_2d

#### *Attributes*

- none

#### *Compositions*

- transformation_target: axis2_placement_2d[1]The transformation_target specifies the axis2_placement_2d defining the target of the transformation.
- transformation_origin: axis2_placement_2d[1]The transformation_origin specifies the axis2_placement_2d defining the origin of the transformation.

#### *Associations*

- none

### 8.5.1.27 Class Implicit_transformation_3d

A geometric relationship between external models can be defined implicitly by using an Implicit_transformation_3d that has two reference points to specify origin and target of the transformation.

#### *Base Class*

- Transformation_3d

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- transformation_origin : Axis2_placement_3d [1]
  The transformation_origin specifies the origin of the transformation.
- transformation_target : Transformation_target_select [1]
  The transformation_target specifies the target of the transformation.

### 8.5.1.28 Class Item_shape

An Item_shape is the definition of the shape of a Design_discipline_item_definition, an Item_instance or of a Physical_instance.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- shape_element : Shape_element [0..*]
  The shape_element specifies the shape_element that is part of this Item_shape.
- shape_description_association : Shape_description_association [0..*]
  The shape_description_association specifies the shape_description_association that is associated with this Item_shape.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Item_shape.
- description : String_select [0..1]The description specifies additional information about the Item_shape.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- described_object : Item_information_select [1]
  The described_object specifies the object whose shape the Item_shape defines.

### 8.5.1.29 Class Material

A Material is the substance out of which an item is or can be made.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- material_name : String [1] The material_name specifies the word or group of words by which the Material is referred to.

#### *Compositions*

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Material.
- material_property_association : Material_property_association [0..*]
  The material_property_association specifies the material_property_association in which a property value is assigned to this Material.

#### *Associations*

- described_element : Item_property_select [1..*]
  The described_element specifies the objects the material information is provided for.

### 8.5.1.30 Class Point

A Point is a location in a Cartesian coordinate space. A Point is a type of Detailed_geometric_model_element.

#### *Base Class*

- ·Detailed_geometric_model_element (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.5.1.31 Class Shape_description_association

A Shape_description_association is a mechanism to associate the definition of a shape or of a portion of a shape with a geometric representation.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1] The role specifies the function performed by the referenced model. Where applicable the following values shall be used:

- 'detailed representation': The geometry in the referenced model provides a detailed representation of the shape;
- 'idealized representation': The geometry in the referenced model provides a simplified representation of the shape, e.g., for analysis purposes.

#### *Compositions*

- none

#### *Associations*

- defining_geometry : Shape_definition_select [1]
  The defining_geometry specifies the Geometric_model or the External_model that contains the shape information.

### 8.5.1.32 Class Shape_element

A Shape_element is a portion of shape that has to be identified explicitly to be associated with other information.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- element_name : String [0..1] The element_name specifies the word or group of words by which the Shape_element is referred to.

#### *Compositions*

- change : Change [0..*] The change specifies the change for which this object references a modified object and the corresponding original object.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Shape_element.
- shape_description_association : Shape_description_association [0..*]
  The shape_description_association specifies the shape_description_association that is associated with this Shape_element.
- shape_element_relationship : Shape_element_relationship [0..*]
  The shape_element_relationship specifies the shape_element_relationship that relates the first of the two Shape_element objects.
- description : String_select [0..1]The description specifies additional information about the Shape_element.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- none

### 8.5.1.33 Class Shape_element_relationship

A Shape_element_relationship is a relationship between two Shape_element objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship.

#### *Compositions*

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Shape_element_relationship.
- shape_description_association : Shape_description_association [0..*]
  The shape_description_association specifies the shape_description_association that is associated with this Shape_element_relationship.
- description : String_select [0..1]The description specifies additional information about the Shape_element_relationship.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- related : Shape_element [1] The related specifies the second of the two Shape_element objects related by a Shape_element_relationship.

### 8.5.1.34 Class Template_instance

A Template_instance is a replica of a Geometric_model, of a Styled_model, or of an External_model.

The location of the replica and any additional geometrical transformation to be applied to the replica, i.e., uniform scaling, rotation or mirroring, are specified by the attributes 'transformation' 'origin' and 'target'.

NOTE 1 In the case where the units of the Cartesian coordinate space of the definition are different from the units to be applied to the Template_instance, unit conversion is required.

NOTE 2 In case of length unit conversion this conversion applies in addition to the scale attribute.

EXAMPLE In a technical drawing of a mechanical part with several identical drilling holes, the hole geometry (circle) together with its annotation elements (diameter dimension and centrelines) is defined once. This particular definition is instantiated several times at different locations by corresponding Template_instance objects.

A Template_instance is a type of Detailed_model_element.

#### *Base Class*

- Detailed_model_element (ABS)

#### *Attributes*

- id: string[1]: The id specifies the identifier of the Template_instance.
- scale: double[0..1]: The scale specifies the scaling factor for all Cartesian coordinate directions. The scaling factor shall be positive. If the scaling factor is omitted, it shall be 1.0.

#### *Compositions*

- target: Explicit_transformation[1]
  The target specifies the geometrical transformation applied to the instance. All transformations that can be expressed by an orthonormal 2 × 2 (for 2D) or 3 × 3 (for 3D) matrix can be applied. EXAMPLE Examples for suitable operations are rotation and mirroring.

#### *Associations*

- template_definition : Template_definition_select[1]
  The template_definition specifies the object to be replicated.
- origin : Axis_placement[1] The origin specifies the Axis_placement that defines the origin of the replicated object.

### 8.5.1.35 Class Transformation (ABS)

A Transformation is a geometric transformation composed of translation and rotation. Scaling is not included.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.5.1.36 Class Transformation_2d

A Transformation_2d is the definition of a geometric transformation in 2D space.

A Transformation_2d is a type of Transformation.

#### *Base Class*

- Transformation

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

#### 8.5.1.37 Class Transformation_3d

A Transformation_3d is the definition of a geometric transformation in 3D space.

#### *Base Class*

- Transformation (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.5.2 Interfaces

#### 8.5.2.1 Interface Accuracy_select

This empty interface is realized by the following classes:

- Geometric_model
- External_geometric_model

#### 8.5.2.2 Interface Geometric_or_external_model_select

This empty interface is realized by the following classes:

- Geometric_model
- External_model (ABS)

#### 8.5.2.3 Interface Shape_definition_select

This empty interface is realized by the following classes:

- Geometric_model
- External_geometric_model

#### 8.5.2.4 Interface Shape_information_select

***Compositions***

- shape_description_association : Shape_description_association [0..*]

***Implemented By***

- Shape_element_relationship
- Shape_element

#### 8.5.2.5 Interface Transformation_select

This empty interface is realized by the following class:

- Template_instance
- Geometric_model_relationship_with_transformation

#### 8.5.2.6 Interface Transformation_target_select

This empty interface is realized by the following classes:

- Explicit_transformation_3d
- Axis2_placement_3d

## 8.6 Package Classification

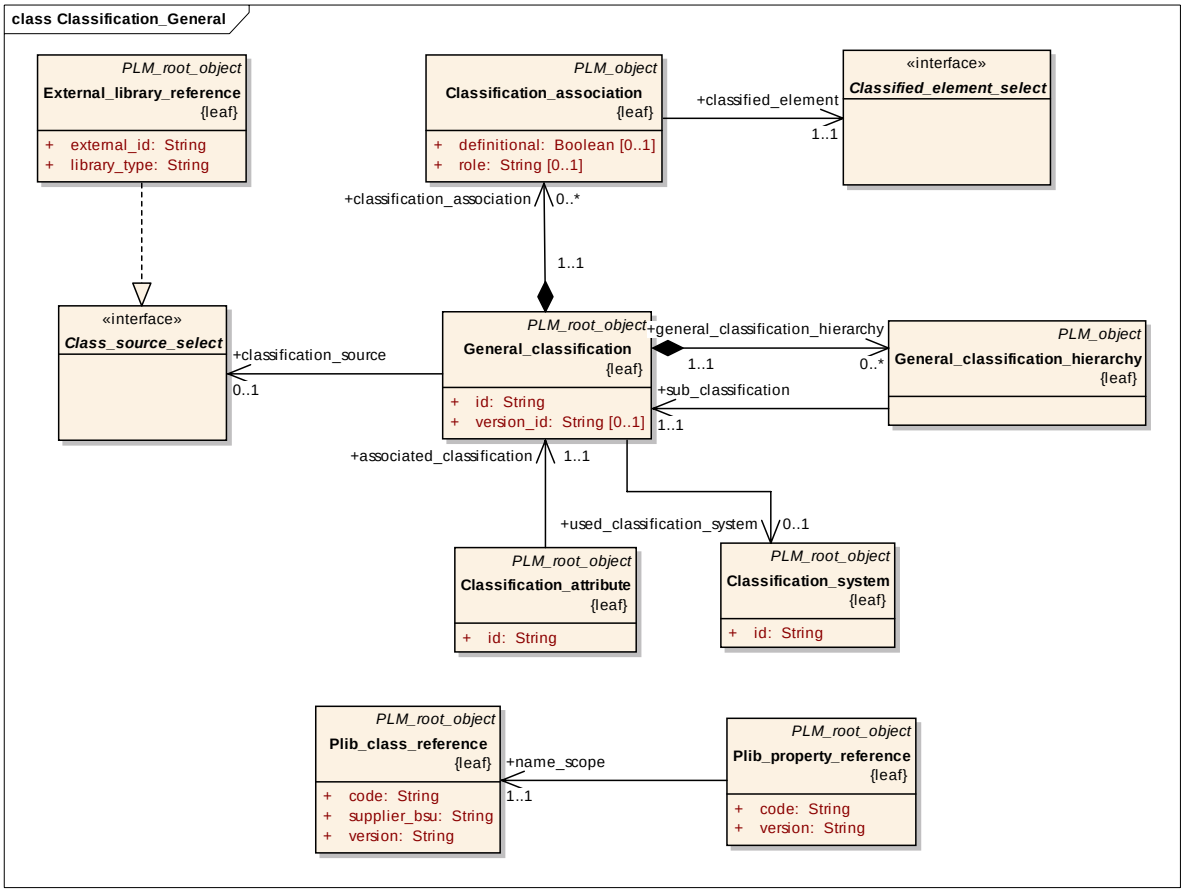


**Figure 8.23 - Classification - General**

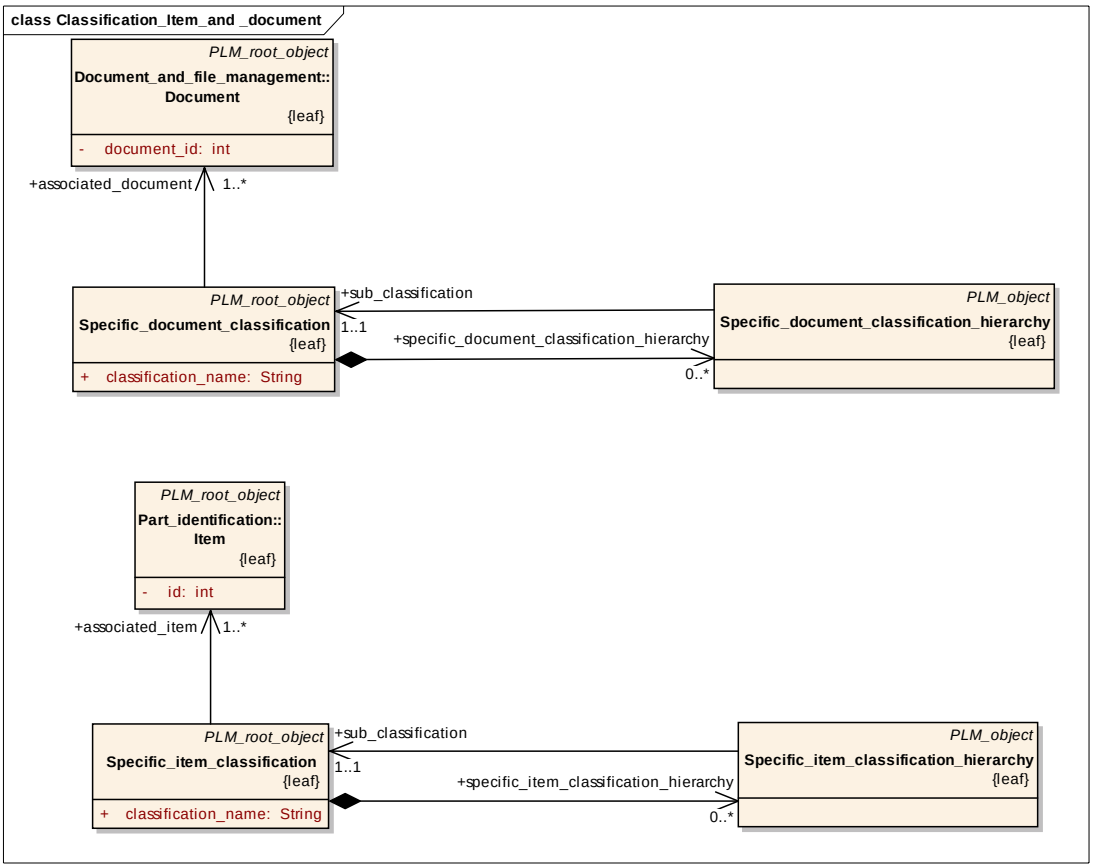


**Figure 8.24 - Classification - Item and document**

## 8.6.1 Classes

### 8.6.1.1 Class Classification_association

A Classification_association associates a General_classification with an object.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [0..1] — The role specifies the relationship between the General_classification and the associated object. Where applicable the following values shall be used:
  - 'electromagnetic compatibility': The associated object is the classification that categorizes the classified element in respect of its ability to comply with requirements concerning electromagnetic interference;
  - 'environmental conditions': The associated object is the classification that categorizes the classified element with respect to its ability to comply with environmental impact requirements.

- definitional : Boolean [0..1] The definitional specifies whether a General_classification serves as definition. A value of 'true' indicates that the General_classification is definitional. The 'associated_classification' does not take precedence over the descriptions of the 'classified_element' made using Property_value or Geometric_model objects.

#### *Compositions*

- none

#### *Associations*

- classified_element : Classified_element_select [1..*]
  The classified_element specifies the objects that are classified.

### 8.6.1.2 Class Classification_attribute

A Classification_attribute is a characteristic used to classify an object associated with the corresponding General_classification. The definition attribute of each 'allowed_value' shall refer to the property identified within 'attribute_definition'.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Classification_attribute that shall be unique within the scope of the associated General_classification.

#### *Compositions*

- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Classification_attribute.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Classification_attribute.
- description : String_select [0..1]The description specifies additional information about the Classification_attribute.
- name : String_select [0..1] The name specifies the word or group of words by which the Classification_attribute is referred to.

#### *Associations*

- attribute_definition : Property (ABS) [1]
  The attribute_definition specifies the Property that characterizes the allowed values.
- allowed_value : Property_value_representation [0..*]
  The allowed_value specifies the set of Property_value_representation objects that represent characteristic values of the Classification_attribute.
- associated_classification : General_classification [1]
  The associated_classification specifies the General_classification the Classification_attribute is a characteristic of.

#### 8.6.1.3 Class Classification_system

A Classification_system is the scheme used to define the categorization of an item.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- id : String [1] The id specifies the identifier of the Classification_system.

***Compositions***

- description : String_select [0..1]The description specifies additional information about the Classification_system.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Classification_system.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Classification_system.

***Associations***

- none

#### 8.6.1.4 Class External_library_reference

An External_library_reference is a mechanism to refer to an entry in an external library other than ISO 13584.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- external_id : String [1] The external_id specifies the unique identifier of the referenced entry in the external library.
- library_type : String [1] The library_type specifies the type of library that is used.

***Compositions***

- description : String_select [0..1]The description specifies additional information about the External_library_reference.

***Associations***

- none

#### 8.6.1.5 Class General_classification

A General_classification is a classification of an object which characterizes all objects of the same kind; such a classification is independent from the application of the classified object.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the General_classification.
- version_id : String [0..1] The version_id specifies the identification of a particular version of the General_classification.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the General_classification.
- general_classification_hierarchy : General_classification_hierarchy [0..*]
  The General_classification_hierarchy specifies the General_classification_hierarchy for which this General_classification is the higher level, and that includes the sub class.
- classification_association : Classification_association [0..*]
  The Classification_association specifies the Classification_association for which this General_classification object provides classification information.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this General_classification.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this General_classification.

#### *Associations*

- classification_source : Class_source_select [0..1]
  The classification_source specifies the External_library_reference or the Plib_class_reference that contains the specification of the General_classification.
- used_classification_system : Classification_system [0..1]
  The used_classification_system specifies the Classification_system that contains the information about the definition of the classification and how to interpret the name of the General_classification.

### 8.6.1.6 Class General_classification_hierarchy

A General_classification_hierarchy defines a hierarchical relationship between two instances of General_classification.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- sub_classification : General_classification [1]
  The sub_classification specifies the lower level of General_classification in a General_classification_hierarchy that is included in the super class.

### 8.6.1.7 Class Plib_class_reference

A Plib_class_reference designates a class in a library compliant to ISO 13584 (Parts Library).

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- code: string[1] The code specifies the class in the PLIB library. The format of this code is defined in ISO 13584-42.
- supplier_bsu: string[1] The supplier_bsu (basic semantic unit) specifies the supplier of the class in a PLIB library, in which the class is defined. The format of this specification is defined in ISO 13584-26.
- version: string[1]: The version specifies the identification of a particular version of a class in a PLIB library. The format of this version is defined in ISO 13584-42.

#### *Compositions*

- none

#### *Associations*

- none

### 8.6.1.8 Class Plib_property_reference

A Plib_property_reference designates a property in a library compliant to ISO 13584.

#### *Base Class*

- ·PLM_root_object (ABS)

#### *Attributes*

- code: string [1] The code specifies the property in the PLIB library. The format of this code is defined in ISO 13584-42.
- version: string [1] The version specifies the identification of a particular version of a property in a PLIB library. The format of this version is defined in ISO 13584-42.

#### *Compositions*

- none

#### *Associations*

- name_scope: Plib_class_reference[1]
  The name_scope specifies the Plib_class_reference in which the property is visible.

#### 8.6.1.9 Class Specific_document_classification

A Specific_document_classification is a classification of a Document with respect to specific criteria. The specific criteria are covered in the 'classification_name' attribute.

##### *Base Class*

- PLM_root_object (ABS)

##### *Attributes*

- classification_name : String [1]The classification_name provides classification information. Where applicable the following values shall be used:
  - 'catalogue': The assigned document is the catalogue in which the associated object is listed;
  - 'manual': The assigned document is the handbook that is supplied for the associated object;
  - 'specification': The assigned document specifies the considerations that lead to the design finally chosen for the associated object.

##### *Compositions*

- specific_document_classification_hierarchy : Specific_document_classification_hierarchy [0..*]
  The Specific_document_classification_hierarchy specifies the Specific_document_classification_hierarchy for which this Specific_document_classification is the higher level, and that is included in the sub class.
- description : String_select [0..1]The description specifies additional information about the Specific_document_classification.

##### *Associations*

- associated_document : Document [1..*]
  The associated_document the associated_document specifies the Document with which a particular Specific_document_classification is associated.

#### 8.6.1.10 Class Specific_document_classification_hierarchy

A Specific_document_classification_hierarchy is used to build up hierarchical structures of Specific_document_classification_hierarchy objects.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- none

##### *Compositions*

- none

#### *Associations*

- sub_classification : Specific_document_classification [1]
  The sub_classification specifies the lower level of Specific_document_classification in Specific_document_classification_hierarchy that is included in the super class.

### 8.6.1.11 Class Specific_item_classification

A Specific_item_classification is a classification of an Item with respect to specific criteria. The specific criteria are covered in the 'classification_name' attribute.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- classification_name : String [1]The classification_name provides high level classification information. Where applicable the following values shall be used:
  - 'application control': This type of classification is used to indicate that an Item shall be considered under certification aspects; these aspects may be specified further by the 'description' attribute;
  - 'assembly': This type of classification shall be used for any Item that has an Assembly_definition provided for at least one of its versions, i.e., it is decomposed further;
  - 'collection': This type of classification shall be used for any Item that has a Collection_definition provided for at least one of its versions;
  - 'completely knocked down': This type of classification is used to indicate that an Item is used in a production site that has assembling facilities only;
  - 'detail': This type of classification shall be used for any Item that has no Assembly_definition provided for any of its versions, i.e., it is not further decomposed;
  - 'in process': This type of classification is used to indicate that the Item identifies an intermediate object in a manufacturing process;
  - 'part': The Item plays the role of a part;
  - 'prototype': This type of classification is used to indicate that the Item identifies a prototype and is not intended for serial production;
  - 'raw material': The Item plays the role of raw material;
  - 'regulated': This type of classification is used to indicate that for an Item certain regulations have to be considered;
  - 'safety': This type of classification is used to indicate that an Item is relevant for safety purposes;
  - 'service': This type of classification is used to indicate that an Item is relevant for service purposes;
  - 'tool': The Item plays the role of a tool.

#### *Compositions*

- specific_item_classification_hierarchy : Specific_item_classification_hierarchy [0..*]
  The Specific_item_classification_hierarchy specifies the Specific_item_classification_hierarchy for which this Specific_item_classification is the higher level, and that includes the sub class.
- description : String_select [0..1]The description specifies additional information about the Specific_item_classification.

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Specific_item_classification.

#### *Associations*

- associated_item : Item [1..*] The associated_item specifies the Item with which a particular Specific_item_classification is associated.

#### 8.6.1.12 Class Specific_item_classification_hierarchy

A Specific_item_classification_hierarchy is used to build up hierarchical structures of Specific_item_classification.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- sub_classification : Specific_item_classification [1]
  The sub_classification specifies the lower level of Specific_item_classification in a Specific_item_classification_hierarchy that is included in the super class.

### 8.6.2 Interfaces

#### 8.6.2.1 Interface Class_source_select

This empty interface is realized by the following class:

- External_library_reference
- Plib_class_reference

#### 8.6.2.2 Interface Classified_element_select

This empty interface is realized by the following classes:

- Design_constraint
- Item
- Approval_status
- Security_level
- Product_class
- Document

- Contract
- Work_request
- Work_order
- Project
- Activity_method
- Activity
- Effectivity
- Specification
- Specification_category
- Product_identification
- Product_class
- Design_constraint
- Complex_product (ABS)
- Document_version
- Document_representation (ABS)
- Document_file (ABS)
- Document
- Item_version
- Design_discipline_item_definition
- Item_instance (ABS)
- Process_plan
- Process_operation_occurrence
- Process_operation_definition
- Property_value_association (ABS)
- Property (ABS)
- Simple_property_association
- Shape_element
- Material

## 8.7 Package Properties

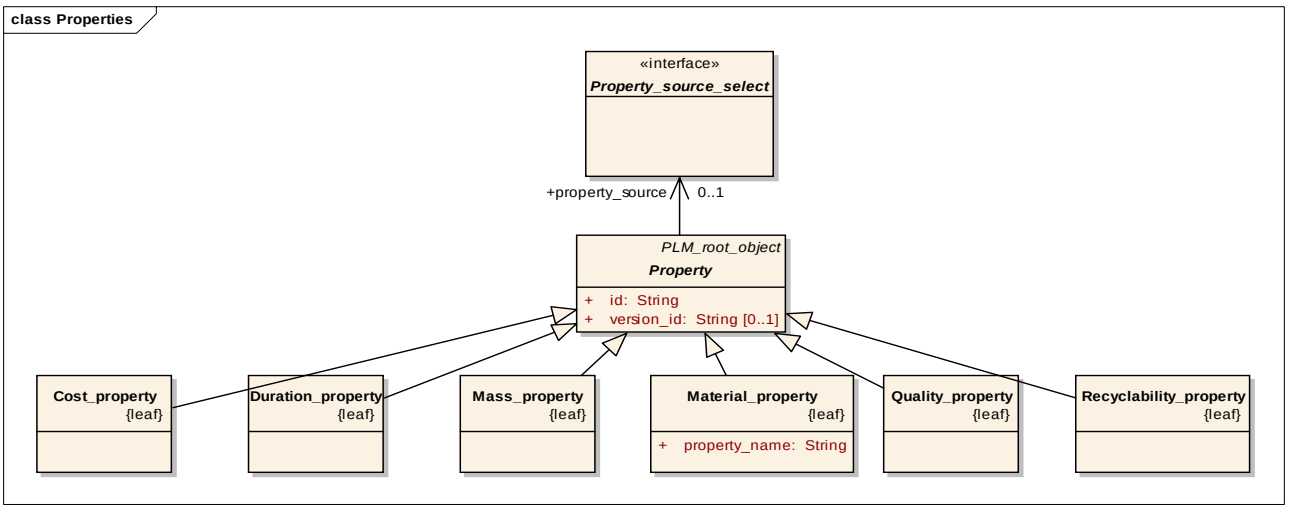


**Figure 8.25 - Properties**

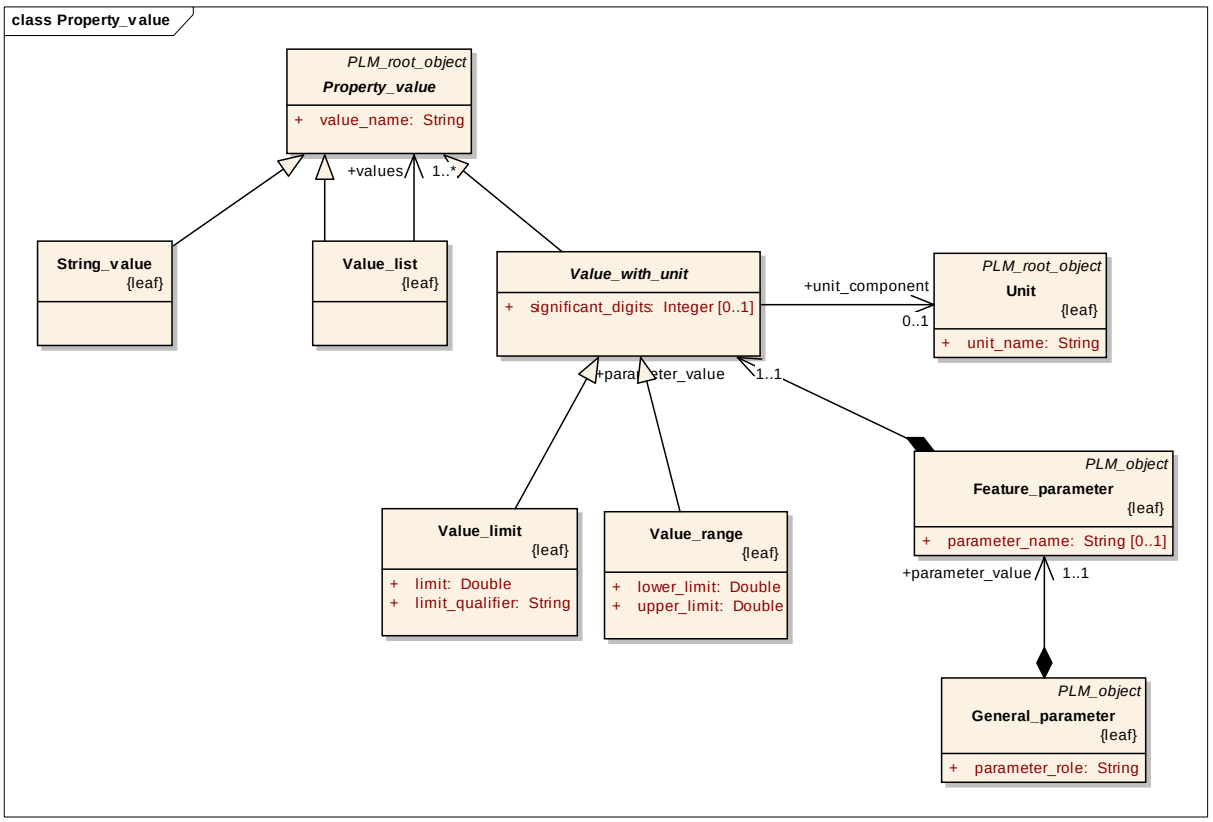


**Figure 8.26 - Property value**

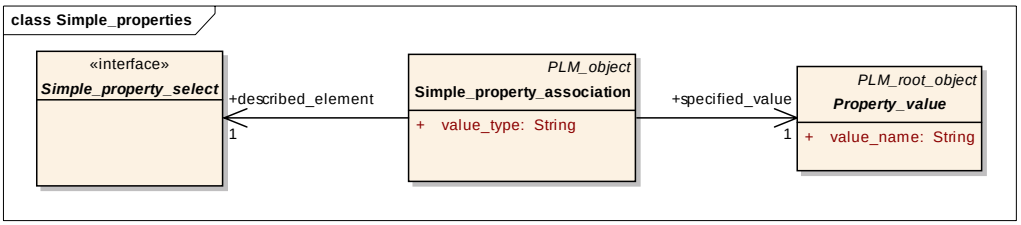


**Figure 8.27 - Simple property**

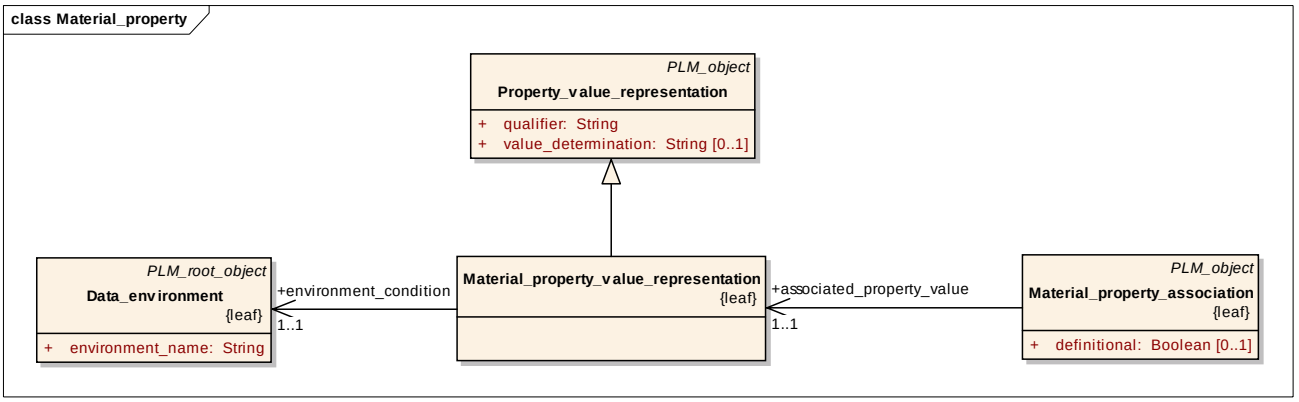


**Figure 8.28 - Material property**

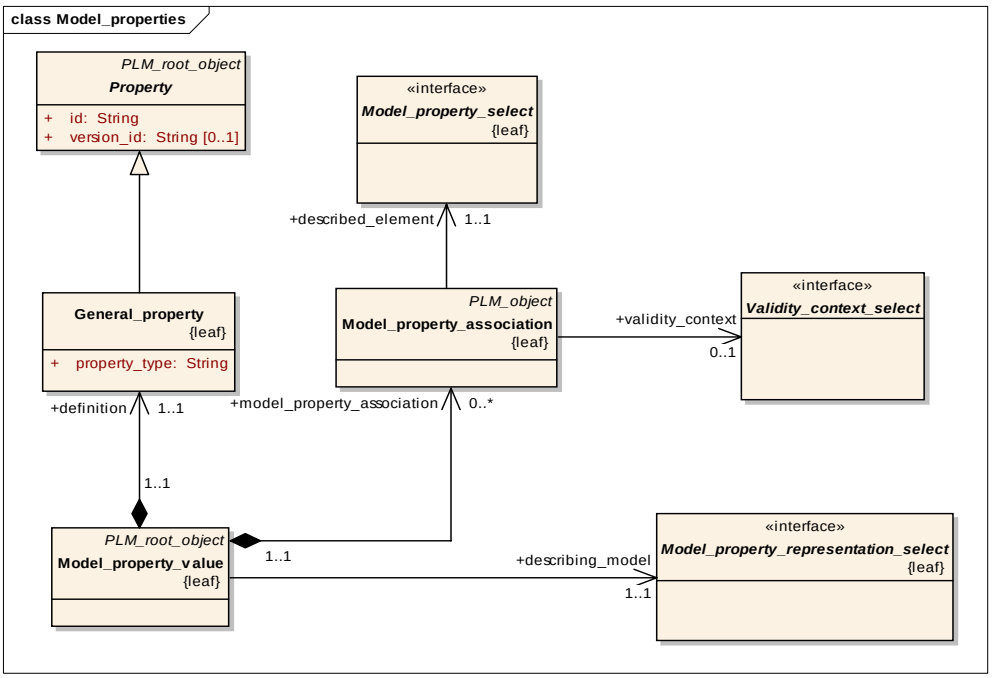


**Figure 8.29 - Model properties**

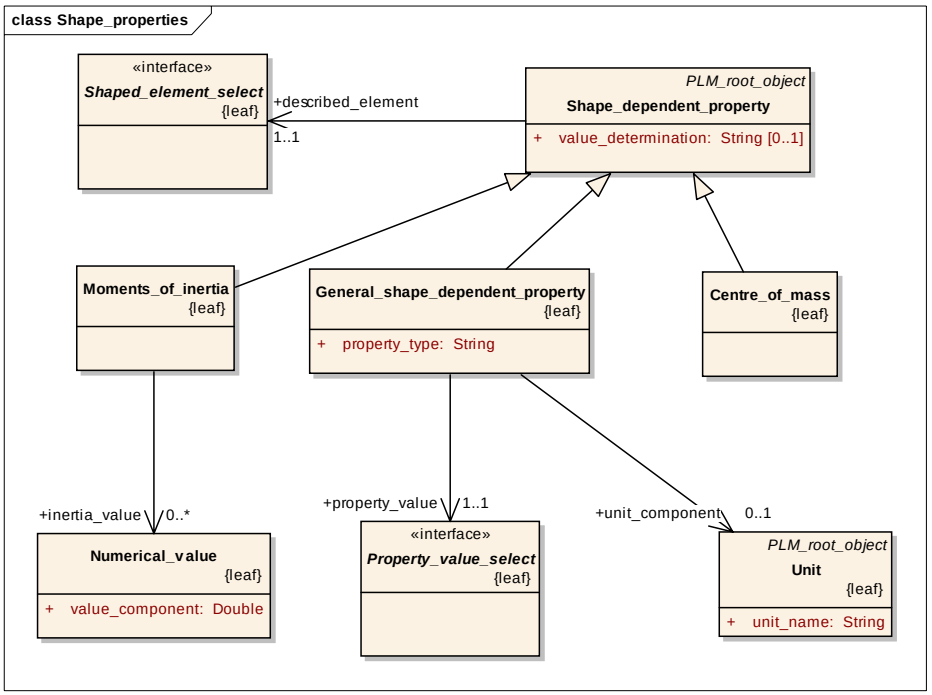


**Figure 8.30 - Shape properties**

## 8.7.1 Classes

### 8.7.1.1 Class Centre_of_mass

A Centre_of_mass is the centre of the mass of a body. The relative position of the Centre_of_mass within the body is an invariant datum relative to rotation and translation. The Centre_of_mass is a characteristic of an item, not its shape.

NOTE 1 If a shape representation is present, the Centre_of_mass is usually derivable from the shape representation. However, as a characteristic of an item, it may be pre-defined in order to reflect a requirement.

NOTE 2 In order to convey the information about the actual centre of mass of a geometric model, the use of General_shape_dependent_property with value 'centroid' for its attribute 'property_type' is appropriate.

A Centre_of_mass is a type of Shape_dependent_property.

#### *Base Class*

- Shape_dependent_property (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- centre_point: Cartesian_point[1]
  The centre_point specifies a point in three-dimensional space that defines the location of a Centre_of_mass in the Cartesian_coordinate_space.

### 8.7.1.2 Class Cost_property

A Cost_property is a property that specifies costs.

#### *Base Class*

- Property (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.3 Class Data_environment

A Data_environment is the specification of the conditions under which a Material_property_value_representation is valid.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- environment_name : String [1] The environment_name specifies the word or group of words by which the Data_environment is referred to.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Data_environment.

#### *Associations*

- none

### 8.7.1.4 Class Duration_property

A Duration_property is a property that specifies a period of time during which a given object is used or will last.

#### *Base Class*

- Property (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.5 Class Feature_parameter

A Feature_parameter is the representation of a characteristic of a Feature_definition.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- parameter_name: string[0..1] The parameter_name specifies the character, abbreviation, or word by which the Feature_parameter is referred to.EXAMPLE 'a', 'b1', and 'r' are typical examples for the parameter_name. The parameter_name need not be specified for a particular Feature_parameter.

#### *Compositions*

- parameter_value: value_with_unit[1]:
  The parameter_value specifies the value of the Feature_parameter. NOTE This includes information about units and possibly about limitations such as tolerances.

#### *Associations*

- none

### 8.7.1.6 Class General_parameter

A General_parameter is the association of a Feature_parameter with a General_feature in a particular role.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- parameter_role: string[1] The parameter_role specifies the kind of parameter that is represented by the General_parameter. EXAMPLE 'width', 'diameter', or 'font size' are examples for the parameter_role.

#### *Compositions*

- parameter_value: feature_parameter[1]
  The parameter_value specifies the Feature_parameter that has the value and possibly a name of the General_parameter.

#### *Associations*

- none

### 8.7.1.7 Class General_property

A General_property is the definition of a property that is specified by the attribute 'property_type'.

#### *Base Class*

- Property (ABS)

#### *Attributes*

- property_type : String [1] The property_type specifies the kind of property the General_property defines. Where applicable the following values shall be used:
  - 'overall axle distance': The overall axle distance is the distance between the first front axle and the rear most axle of the vehicle combination;
  - 'positioning': The General_property is the definition of a Model_property_value that provides an a geometric model for a Product_component or an Item_instance for the purpose of placement;
  - 'theoretical wheelbase': The theoretical wheelbase is the distance between the resolved weight lines of front and rear axle combinations;
  - 'track': The track is the distance between the centre of the tires mounted on an axle of a vehicle;
  - 'wheel space': The wheel space is the distance between the perpendicular lines constructed to the longitudinal median plane of the vehicle from two points that represent the wheels situated at the same side of the axle that is of interest.

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.8 Class General_shape_dependent_property

A General_shape_dependent_property is a user-defined characteristic of an object. The 'property_value' specified by a General_shape_dependent_property is derived from geometry.

NOTE The General_shape_dependent_property may be used for the purpose of validation of geometric models.

EXAMPLE If a geometry model is exchanged between two partners, the calculation of various properties of the bodies, such as centre of mass, overall surface, or overall volume, is performed by originator and recipient. When comparing these set of values, the existence of deficiencies in the received model can be easily detected.

A General_shape_dependent_property is a type of Shape_dependent_property.

#### *Base Class*

- Shape_dependent_property (ABS)

#### *Attributes*

- property_type : string[1] The property_type defines the type of characteristic that is specified for an object.

#### *Compositions*

- none

#### *Associations*

- property_value : Property_value_select[1]
  The property_value specifies the value that is given for a particular characteristic.
- unit_component : Unit[0..1] The unit_component specifies the unit in which the General_shape_dependent_property is expressed.

### 8.7.1.9 Class Item_property_association

An Item_property_association is a mechanism to associate a property value with an object.

#### *Base Class*

- Property_value_association (ABS)

#### *Attributes*

- definitional : Boolean [0..1] The definitional specifies whether the associated Property_value_representation object may be used to distinguish the described_element from others of the same kind. A value of 'true' indicates that the associated Property_value_representation distinguishes it from others.

#### *Compositions*

- none

#### *Associations*

- described_element : Item_property_select [1]
  The described_element specifies the object that is characterized by the Property_value.

### 8.7.1.10 Class Mass_property

A Mass_property is a quantity of matter that an object consists of.

#### *Base Class*

- Property (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.11 Class Material_property

A Material_property is a characteristic that depends on material aspects.

#### *Base Class*

- Property (ABS)

#### *Attributes*

- property_name : String [1] The property_name specifies the kind of Material_property.

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.12 Class Material_property_association

A Material_property_association is an object that associates a Material object with a Material_property_value_representation object.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- definitional : Boolean [0..1] The definitional specifies whether the associated_property_value may be used to distinguish the described_material from others of the same kind. A value of 'true' indicates that the Material_property_value_representation distinguishes the 'described_element' from others.

#### *Compositions*

- none

#### *Associations*

- associated_property_value : Material_property_value_representation [1]
  The associated_property_value specifies the associated Material_property_value_representation.

### 8.7.1.13 Class Material_property_value_representation

A Material_property_value_representation is the representation of a characteristic of a material.

#### *Base Class*

- Property_value_representation

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- environment_condition : Data_environment [1]
  The environment_condition specifies the environmental conditions in which the defining Material_property_value_representation is applicable.

### 8.7.1.14 Class Model_property_association

A Model_property_association is an association of a Model_property_value with objects.

EXAMPLE In the case of a welding operation, the welding points as a part of a wireframe model can be associated with a process operation to describe the locations where the robot has to weld for a welding process operation.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- described_element: Model_property_select[1]
  The described_element specifies the application object for which a Model_property_value is provided.
- validity_context: Validity_context_select[0..1]
  The validity_context specifies the context in which a Model_property_association is applicable.

### 8.7.1.15 Class Model_property_value

A Model_property_value is a mechanism to assign a General_property to an External_model, a Geometric_model, or a Styled_model.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- definition : general_property[1]The definition specifies the General_property that is assigned to an External_model, a Geometric_model, or a Styled_model.
- model_property_assocation : model_property_assocation[0..*]

#### *Associations*

- describing_model : Model_property_representation_select[1]
  The describing_model specifies the model to which a General_property is assigned.

### 8.7.1.16 Class Moments_of_inertia

A Moments_of_inertia describes the matrix of inertia of a rigid body. The moments of inertia I are described as follows

$$I = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{pmatrix}$$

A Moments_of_inertia is a type of Shape_dependent_property.

#### *Base Class*

- Shape_dependent_property (ABS)

#### *Attributes*

- none

#### *Compositions*

- reference_axis_placement : axisplacement[1]
  The reference_axis_placement specifies the Axis_placement which was used to calculate the 'inertia_value'.

#### *Associations*

- inertia_value : Numerical_value[3][3]
  The inertia_value specifies the 9 inertia coefficients. NOTE The matrix defined by the 'inertia_value' is symmetric with respect to the main diagonal. For that reason, the values of the coefficients are equal, if mirrored along this diagonal. As a consequence, there are 6 different values in the 'inertia_value'.

### 8.7.1.17 Class Numerical_value

A Numerical_value is a quantity expressed with a numerical value and a unit.

#### *Base Class*

- Value_with_unit (ABS)

#### *Attributes*

- value_component : Double [1] The value_component specifies the quantity of the Numerical_value.

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.18 Class Property (ABS)

A Property is the definition of a particular quality.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Property.
- version_id : String [0..1] The version_id specifies the identification of a particular version of a Property.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Property.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Property.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Property.

#### *Associations*

- property_source : Property_source_select [0..1]
  The property_source specifies the External_library_reference or Plib_property_reference object that defines this kind of property.
- allowed_unit : Unit [0..*] The allowed_unit specifies the unit or set of units that are accepted.

### 8.7.1.19 Class Property_change

A Property_change is a mechanism to describe the differences between two objects concerning the properties of these objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- id: string[0..1] The id specifies the identifier of the Property_change. The id need not be specified for a particular Property_change.

#### *Compositions*

- description: String_select[1] The description specifies additional information about the Property_change.

#### *Associations*

- is_describing: Change[1]: The is_describing specifies the Change that collects the Property_change objects and the Model_change objects.
- added_property: Property_value_representation[0..*]
  The added_property specifies the set of Property_value_representation objects that have been added to that object of the described relationship that is identified as relating.
- deleted_property: Property_value_representation[0..*]
  The deleted_property specifies the Property_value_representation objects that have been deleted from that object of the described relationship that is identified as relating.

### 8.7.1.20 Class Property_relationship

A Property_relationship is a relationship between two Property objects.

EXAMPLE Property_relationship may be used to indicate that the value of one Property can be derived from the value of another Property.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type: string[1]: The relation_type specifies the meaning of the relationship.

#### *Compositions*

- description: String_select[0..1]: The description specifies additional information about the Property_relationship.

#### *Associations*

- related : Property[1] The related specifies the second of the two Property objects related by the Property_relationship. NOTE The semantics of this attribute are defined by the attribute relation_type.

### 8.7.1.21 Class Property_value (ABS)

A Property_value is the numerical or textual value of a Property_value_representation.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- value_name : String [1] The value_name specifies the word or group of words by which the Property_value is referred to.

#### *Compositions*

- property_value_representation : Property_value_representation [0..*]
  The property_value_representation specifies the property_value_representation that is qualified by this Property_value, by a Value_with_unit, a String_value, or an arbitrary aggregate thereof.

#### *Associations*

- none

### 8.7.1.22 Class Property_value_association (ABS)

A Property_value_association is a mechanism to assign a Property_value_representation to an object.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Property_value_association.

#### *Associations*

- validity_context : Validity_context_select [0..1]
  The validity_context specifies the context in which a Property_value_association is applicable.

### 8.7.1.23 Class Property_value_representation

A Property_value_representation is the representation of Property.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- value_determination : String [0..1]The value_determination specifies information on how the Property_value_representation shall be interpreted. Where applicable the following values shall be used:
  - 'calculated': The value has been calculated;
  - 'designed': The value represents a value intended by the design;
  - 'estimated': The value has been estimated;
  - 'measured': The value has been measured;
  - 'required': The value represents a requirement;
  - 'set point': The value is used as the initialization value.
- qualifier : String [0..1] The qualifier specifies the kind of the Property_value_representation. The following values shall be used:

- 'nominal': The value is the nominal value;
- 'specified': The value is specified;
- 'typical': The value is a typical value.

#### *Compositions*

- property_value_association : Property_value_association (ABS) [0..*]
  The property_value_association specifies the property_value_association which this object is assigned to.

#### *Associations*

- definition : Property (ABS) [1] The definition specifies the Property that the Property_value_representation characterizes. If the Property_value_representation is a Material_property_value_representation, the definition shall specify a Material_property.
- global_unit : Unit [0..1] The global_unit specifies a unit that is valid for all Property_value that are referenced as 'specified_value' by the Property_value_representation.

### 8.7.1.24 Class Property_value_representation_relationship

A Property_value_representation_relationship is a relationship between two Property_value objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type: string[1] The relation_type specifies the meaning of the relationship.
  EXAMPLE Property_value_representation objects representing 'cycle time', 'feed time', and 'drill time' for a process may be related by a Property_value_representation_relationship with relation_type 'dependency'.

#### *Compositions*

- description: String_select[0..1]
  The description specifies additional information about the Property_value_representation_relationship.

#### *Associations*

- related : Property_value_representation[1]
  The related specifies the second of the two objects related by the Property_value_representation_relationship. NOTE The semantics of this attribute is defined by the attribute relation_type.

### 8.7.1.25 Class Quality_property

A Quality_property is a property that enables to provide information about the level of quality of products or processes.

#### *Base Class*

- Property (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.26 Class Recyclability_property

A Recyclability_property is information concerning the ability to reuse objects or components of objects after their primarily intended usage.

#### *Base Class*

- Property (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.27 Class Shape_dependent_property

A Shape_dependent_property is a characteristic of the shape, or of a portion of the shape of an object. This object is either an item, an occurrence of an item in the product structure or the physical realization of an item.

NOTE A Shape_dependent_property is independent of the representations of the shape.

Each Shape_dependent_property is a Centre_of_mass, a Moments_of_inertia, or a General_shape_dependent_property.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- value_determination: string[0..1]

  The value_determination specifies information on how the Shape_dependent_property shall be interpreted. NOTE A Shape_dependent_property may be specified in the design stage of a product but it may also be documented as measured on a prototype. The value_determination need not be specified for a particular Shape_dependent_property.

### *Compositions*

- description: String_select[0..1]
  The description specifies additional information about the Shape_dependent_property.

### *Associations*

- described_element : Shaped_element_select[1]
  The described_element specifies the object that the Shape_dependent_property is associated to.
- is_defined_in : Cartesian_coordinate_space[0..1]
  The is_defined_in specifies the Cartesian_coordinate_space in which the property is applicable. NOTE The values of geometry related properties are specified relative to a Cartesian_coordinate_space. EXAMPLE A Centre_of_mass may be specified in different coordinate spaces for different dimensioning applications.

#### 8.7.1.28 Class Simple_property_association

A Simple_property_association holds a type and relates a property value by one of the subtypes of property_value to an instance of simple_property_select.

### *Base Class*

- PLM_object (ABS)

### *Attributes*

- value_type : String [1]
  The property_type specifies the kind of property the Property_value defines. Where applicable the following values shall be used:
  - 'cost': The cost of an object;
  - 'duration': The duration specifies a period of time during which a given object is used or will last;
  - 'mass': The mass is the quantity of matter that an object consists of;
  - 'overall axle distance': The overall axle distance is the distance between the first front axle and the rear most axle of the vehicle combination;
  - 'positioning': The General_property is the definition of a Model_property_value that provides an a geometric model for a Product_component or an Item_instance for the purpose of placement;
  - 'quality': The quality of products or processes;
  - 'recyclability': The recyclability is the ability to reuse objects or components of objects after their primarily intended usage;
  - 'theoretical wheelbase': The theoretical wheelbase is the distance between the resolved weight lines of front and rear axle combinations;
  - 'track': The track is the distance between the centre of the tyres mounted on an axle of a vehicle;
  - 'wheel space': The wheel space is the distance between the perpendicular lines constructed to the longitudinal median plane of the vehicle from two points that represent the wheels situated at the same side of the axle that is of interest.

### *Compositions*

- none

#### *Associations*

- specified_value : Property_value[1]
  The specified_value denotes the concrete subtype instance of Property_value this simple_property_association relates to the described simple_property_select.

### 8.7.1.29 Class String_value

A String_value represents a sequence of one or more alphanumeric characters.

#### *Base Class*

- Property_value (ABS)

#### *Attributes*

- none

#### *Compositions*

- value_specification : String_select [1]
  The value_specification specifies the string represented by the String_value.

#### *Associations*

- none

### 8.7.1.30 Class Unit

A Unit is a quantity chosen as a standard in terms of which other quantities may be expressed.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- unit_name : String [1] The unit_name specifies the term representing the kind of unit.

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.31 Class Value_limit

A Value_limit is a qualified numerical value representing either the lower limit or the upper limit of a particular physical characteristic.

#### *Base Class*

- Value_with_unit (ABS)

#### *Attributes*

- limit_qualifier : String [1] The limit_qualifier specifies the kind of limit.
- limit : Double [1] The limit specifies the value of the limit.

#### *Compositions*

- none

#### *Associations*

- none

### 8.7.1.32 Class Value_list

A Value_list is an ordered collection of Property_value objects.

#### *Base Class*

- Property_value (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- values : Property_value (ABS) [1..*]
  The values specifies the ordered collection of Property_value objects that together are provided as a Property_value.

### 8.7.1.33 Class Value_range

A Value_range is a pair of numerical values representing the range in which the value shall lie.

#### *Base Class*

- Value_with_unit (ABS)

#### *Attributes*

- upper_limit : Double [1] The upper_limit specifies the maximum acceptable value that is constrained by the Value_range.
- lower_limit : Double [1] The lower_limit specifies the minimum acceptable value that is constrained by the Value_range.

#### *Compositions*

- none

#### *Associations*

- none

#### 8.7.1.34 Class Value_with_unit (ABS)

A Value_with_unit is either a single numerical measure, or a range of numerical measures with upper, lower, or upper and lower bounds.

#### *Base Class*

- Property_value (ABS)

#### *Attributes*

- significant_digits : Integer [0..1] The significant_digits specifies the number of decimal digits that are relevant for the use of the Value_with_unit. If present, the numerical measure or range may be specified using more digits than the significant digits but shall not be specified using less digits.

#### *Compositions*

- none

#### *Associations*

- unit_component : Unit [0..1] The unit_component specifies the unit in which the Value_with_unit is expressed.

### 8.7.2 Interfaces

#### 8.7.2.1 Interface Item_property_select

This empty interface is realized by the following classes:

- Product_structure_relationship
- Product_identification
- Product_class
- Physical_instance
- Design_constraint
- Complex_product (ABS)
- Document_representation (ABS)
- Document_file (ABS)
- Item_definition_relationship (ABS)
- Design_discipline_item_definition
- Item_instance_relationship (ABS)
- Item_instance (ABS)
- Item_definition_instance_relationship (ABS)
- Shape_element_relationship

- Shape_element
- Item_shape

#### 8.7.2.2 Interface Simple_property_select

***Compositions***

- simple_property_association : Simple_property_association [0..*]

***Extended by***

- Item_property_select
- Process_property_select

#### 8.7.2.3 Interface Property_source_select

This empty interface is realized by the following class:

- External_library_reference
- Plib_property_reference

#### 8.7.2.4 Interface Validity_context_select

This empty interface is realized by the following classes:

- Organization
- Product_identification
- Product_class

## 8.8 Package Alias_identification

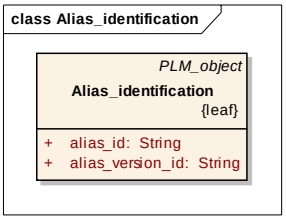


**Figure 8.31 - Alias identification**

### 8.8.1 Classes

#### 8.8.1.1 Class Alias_identification

An Alias_identification is a mechanism to associate an object with an additional identifier that is used to identify the object of interest in a different context, either in another Organization, or in some other context. The scope of the Alias_identification shall be specified either by the attribute 'alias_scope' or by the attribute 'description'.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- alias_id : String [1] The alias_id specifies the identifier used in the context specified by the alias_scope, or by the description.
- alias_version_id : String [0..1] The alias_version_id specifies the version of the object as known in the context of the Alias_identification.

#### *Compositions*

- description : String_select [0..1]The description specifies the type of the Alias_identification.

#### *Associations*

- alias_scope : Organization [0..1]The alias_scope specifies the Organization in which the Alias_identification is valid.

### 8.8.2 Interfaces

#### 8.8.2.1 Interface Alias_select

#### *Compositions*

- alias_identification : Alias_identification [0..*]

#### *Implemented By*

- Organization
- Complex_product
- Classification_attribute
- Item
- Document_type_property
- Product_class
- Document_version
- Specification_category
- Document
- Specification
- Security_level
- Item_version
- Classification_system
- Item_instance
- Document_representation
- Geometric_model

- Property
- General_classification
- Design_discipline_item_definition
- Physical_instance
- Approval_status

## 8.9 Package Authorization

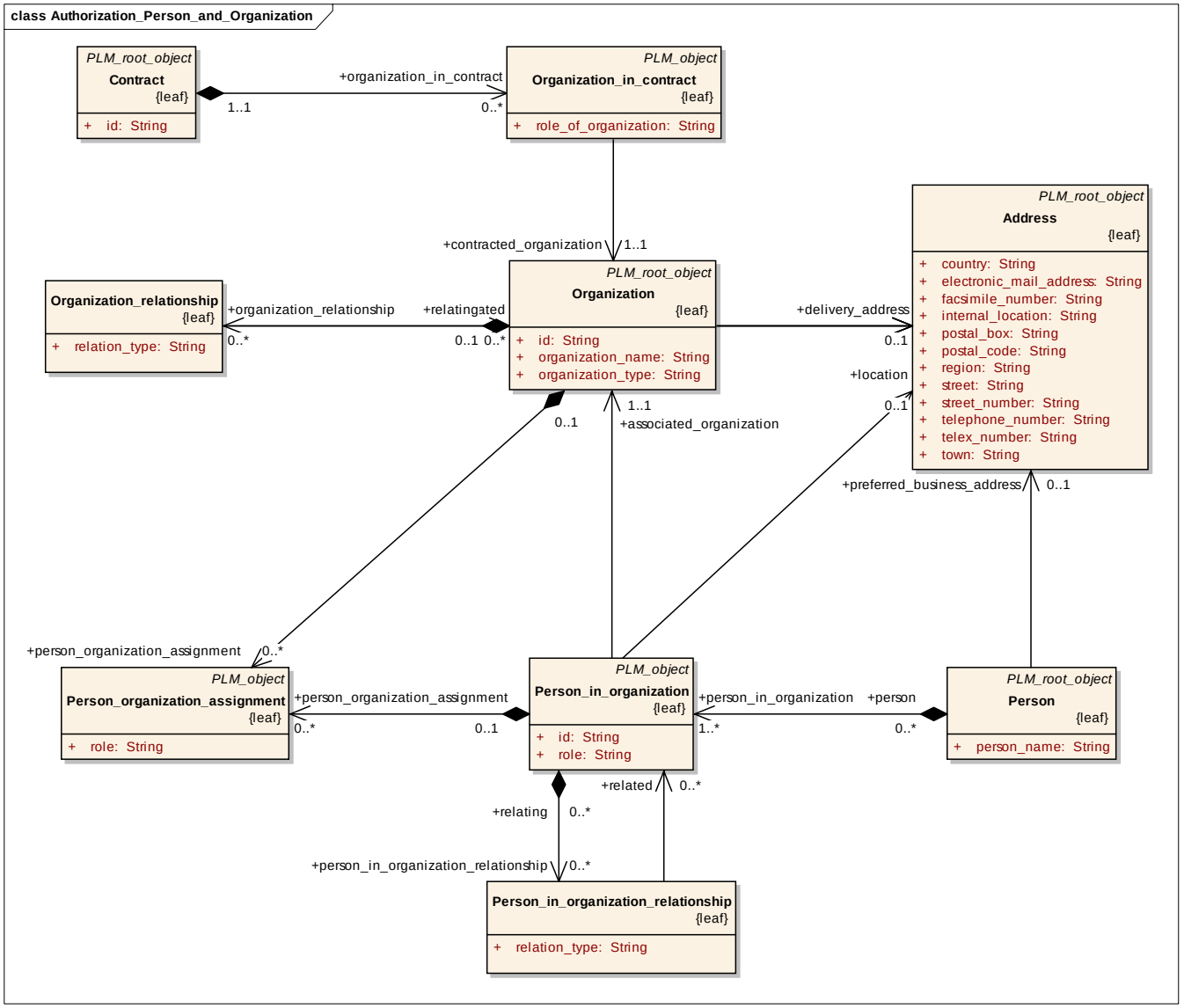


**Figure 8.32 - Authorization - Person and organization**

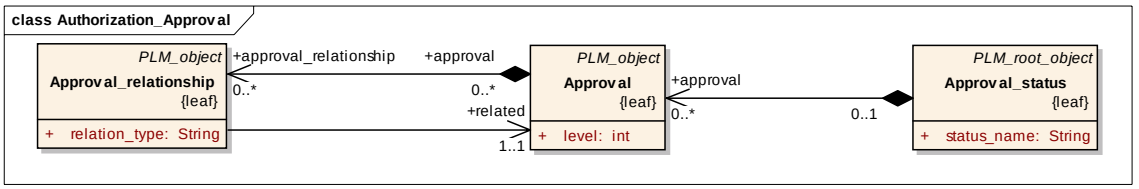


**Figure 8.33 - Authorization - Approval**

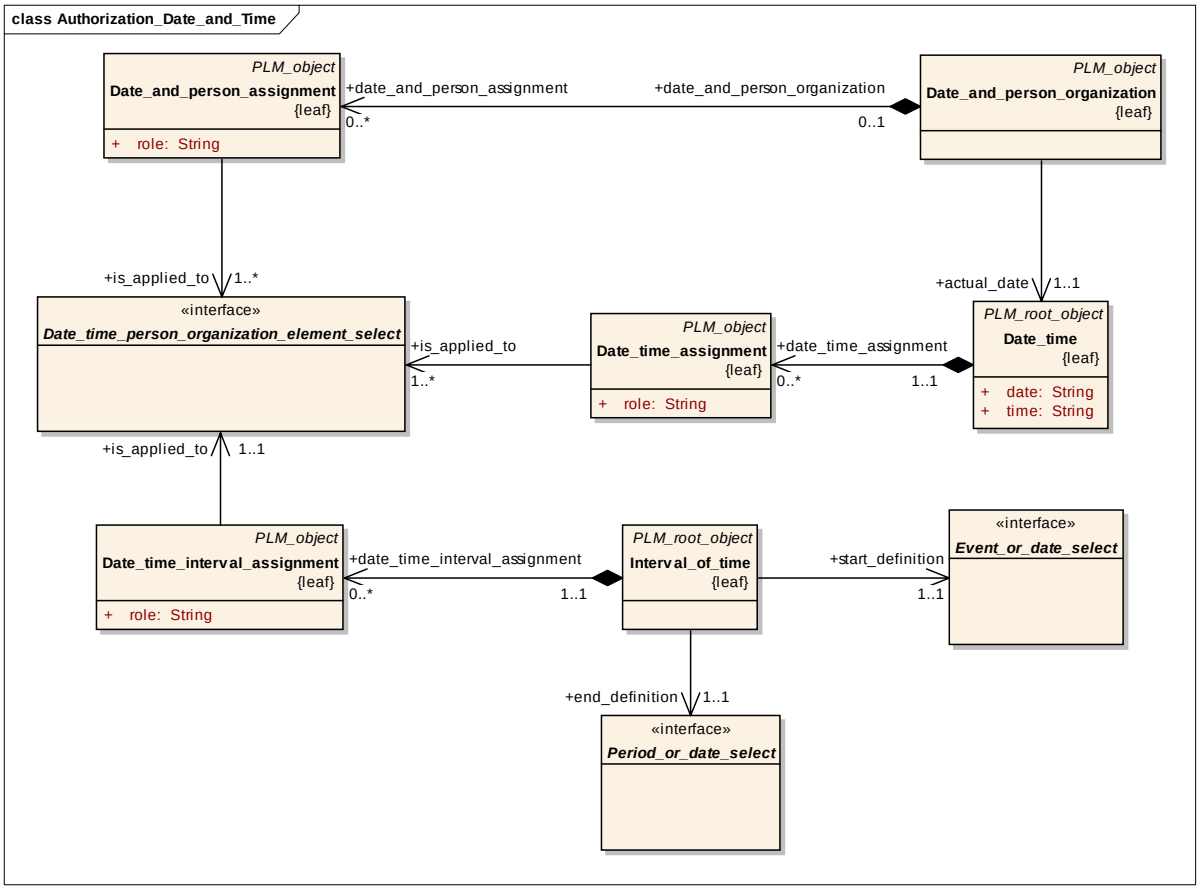


**Figure 8.34 - Authorization - Date and time**

## 8.9.1 Classes

### 8.9.1.1 Class Address

An Address contains information about how a person or an organization can be contacted.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- internal_location : String [0..1] The internal location.
- street_number : String [0..1] The street number.
- street : String [0..1] The street.
- postal_box : String [0..1] The postal box.
- town : String [0..1] The town.
- region : String [0..1] The region.
- postal_code : String [0..1] The postal code.
- country : String [0..1] The country.
- facsimile_number : String [0..1]The fax number.
- telephone_number : String [0..1]The telephone number.
- electronic_mail_address : String [0..1]
  The e-mail address.
- telex_number : String [0..1] The telex number.

#### *Compositions*

- none

#### *Associations*

- none

### 8.9.1.2 Class Approval

An Approval is a judgement concerning the quality of those product data that are subject of the Approval. An Approval represents a statement made by technical personnel or management personnel whether certain requirements are met. The absence of approval information does not imply any approval status by default.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- level : String [0..1] The level represents the aspect for which the object subject to approval, by reference as 'is_applied_to', is endorsed. Where applicable the following values shall be used:
  - 'disposition': The referenced object is approved for series production;
  - 'equipment order': The referenced object has reached a status in which changes are subject to a defined change process and tools and other equipment required for production may be ordered;
  - 'planning': The referenced object is technically complete and has reached a status sufficiently stable so that other designs may be based on it.

#### *Compositions*

- approval_relationship : Approval_relationship [0..*]
  The Approval_relationship specifies the Approval_relationship that relates the first of the two Approval objects.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Approval.

#### *Associations*

- scope : Organization [0..*] The scope specifies the set of Organization objects for which the Approval is valid.
- actual_date : Date_time [0..1] The actual_date specifies the date when the Approval actually became valid. If this attribute is absent, the approval has not yet occurred, i.e., it is pending.
- planned_date : Date_time [0..1]The planned_date specifies the date when the Approval is or was supposed to be performed.
- is_approved_by : Date_and_person_organization [0..*]
  The is_approved_by specifies personnel responsible for the Approval and the dates of the Approval.
- is_applied_to : Approval_element_select [1..*]
  The is_applied_to specifies the objects to which the Approval is assigned.

### 8.9.1.3 Class Approval_relationship

An Approval_relationship is a relationship between two Approval objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'decomposition': The Approval_relationship defines a relationship where the related Approval is one of the components into which the relating Approval is broken down with no implication of 'sequence' or 'dependency';
  - 'dependency': The Approval_relationship defines a relationship where the issuing of the related Approval is dependent on the issuing of the relating Approval;
  - 'precedence': the Approval_relationship defines a relationship where the related Approval has higher priority than the relating Approval;
  - 'sequence': The Approval_relationship defines a relationship where the relating Approval shall be completed before the related Approval is given.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Approval_relationship.

#### *Associations*

- related : Approval [1] The related specifies the second of the two Approval objects related by the Approval_relationship.

### 8.9.1.4 Class Approval_status

An Approval_status is the state of acceptance of some product data.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- status_name : String [1] The status_name specifies the terms characterizing the Approval_status.

#### *Compositions*

- approval : Approval [0..*] The Approval indicates the approval that is applied to the level of acceptance of this Approval_status, for the specified 'level'.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Approval_status.

#### *Associations*

- used_classification_system : Classification_system [0..1]
  The used_classification_system specifies the Classification_system that contains the informa-tion about how to interpret the Approval_status.

### 8.9.1.5 Class Certification

A Certification is a certificate for an object.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- certification_type: string[1]
  The certification_type specifies the kind of certification. EXAMPLE 'supplier certificate' is an example for the certification_type.

#### *Compositions*

- name: String_select[1]: The name specifies the word or group of words by which the Certification is referred to.
- purpose: String_select[0..1]: The purpose specifies the objective of the Certification.

#### *Associations*

- is_applied_to: Certification_select [1]
  The is_applied_to specifies the set of objects that the certificate is applied to.

#### 8.9.1.6 Class Contract

A Contract is a binding agreement concerning the design of Item_version objects, the delivery of Drawing objects, or the execution of other Activity objects.

##### *Base Class*

- PLM_root_object (ABS)

##### *Attributes*

- id: string[1] The id specifies the identifier of the Contract that shall be unique within the scope of an organization.

##### *Compositions*

- description: String_select[0..1]:
  The description specifies additional information for an Item_version, a Drawing, or an Activity object according to the underlying Contract. EXAMPLE The description of the Contract may be the ordered price.
- organization_in_contract: organization_in_contract[0..*]

##### *Associations*

- contracted_element: Contracted_element_select[0..*]
  The contracted_element specifies the object that is subject of the Contract.

#### 8.9.1.7 Class Date_and_person_assignment

A Date_and_person_assignment is an object that associates a Date_and_person_organization with product data. This assignment provides additional in-formation for the associated object.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- role : String [1] The role specifies the relationship between the date or time and the person or organization in the Date_and_person_assignment. Where applicable the following values shall be used:
  - 'creation': The assignment specifies that the referenced object has been created by the given person or organization at the given date and time;
  - 'update': The assignment specifies that the referenced object has been altered by the given person or organization at the given date and time.

##### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Date_and_person_assignment.

##### *Associations*

- is_applied_to : Date_time_person_organization_element_select [1..*]
  The is_applied_to specifies the set of objects with which the Date_and_person_assignment is associated.

#### 8.9.1.8 Class Date_and_person_organization

A Date_and_person_organization is a Person_in_organization or an Organization associ-ated with a Date_time or an Event_reference.

***Base Class***

- PLM_object (ABS)

***Attributes***

- none

***Compositions***

- date_and_person_assignment : Date_and_person_assignment [0..*]
  The Date_and_person_assignment specifies the Date_and_person_assignment for this Date_and_person_organization.

***Associations***

- actual_date : Date_time [1] The actual_date specifies the date and an optional time of day component of a Date_and_person_organization, or alternatively a discrete point in time as an Event_reference.

#### 8.9.1.9 Class Date_time

A Date_time is the specification of a date and an optional time of day.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- time : String [0..1] The time specifies a moment of occurrence measured by hour, minute, and second.
- date : String [1] The date specifies the calendar time, defined according to the Gregorian calendar, conveying information about the year, the month, and the day in no specific order. The representation of a date shall be complete, i.e., millennium, century, and year-within-century data shall be included.

***Compositions***

- date_time_assignment : Date_time_assignment [0..*]
  The Date_time_assignment specifies the Date_time_assignment which this Date_time is assigned to.

***Associations***

- none

### 8.9.1.10 Class Date_time_assignment

A Date_time_assignment is an association of point in time specified as a Date_time or an Event_reference with product data.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1] The role specifies the action associated with the Date_time_assignment. Where applicable the following values shall be used:
  - 'classification date': The assignment specifies that the specified object is classified at the given date and time. This value shall only be used, if the Date_time_assignment refers to instances of Classification_association as 'is_applied_to';
  - 'creation': The assignment specifies that the referenced object was created at the given date and time;
  - 'installation': The assignment specifies that the referenced object was mounted in a product at the given date and time;
  - 'production': The assignment specifies that the referenced object was produced at the given date and time;
  - 'registration': The assignment specifies that the referenced object was determined at the given date and time;
  - 'update': The assignment specifies that the referenced object was altered at the given date and time.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Date_time_assignment.

#### *Associations*

- is_applied_to : Date_time_person_organization_element_select [1..*]
  The is_applied_to specifies the set of objects of product data with which the Date_time_assignment is associated.

### 8.9.1.11 Class Date_time_interval_assignment

A Date_time_interval_assignment is an association of an Interval_of_time with some product data. The Date_time_interval_assignment shall neither be used to convey effectivity information, i.e., information concerning valid use of product data, nor to convey retention information, nor duration information.

NOTE Such information is conveyed using Effectivity, Retention_period, and Duration respectively.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role: string[1] The role specifies the action associated with the Date_time_interval_assignment.

#### *Compositions*

- description: String_select[0..1]
  The description specifies additional information about the Date_time_interval_assignment.

#### *Associations*

- is_applied_to: Date_time_person_organization_select[1]
  The is_applied_to specifies the set of objects of product data with which the Date_time_interval_assignment is associated.

### 8.9.1.12 Class Duration

A Duration is the definition of a period of time.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- time : String [1] The time specifies the extend of the Duration.
- time_unit : String [1] The time_unit specifies the unit in which the time is specified.

#### *Compositions*

- none

#### *Associations*

- none

### 8.9.1.13 Class Event_reference

An Event_reference is the definition of a point in time established relative to an event.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- event_type : String [1] The event_type specifies the kind of event that serves as reference.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Event_reference.

#### *Associations*

- event_context : General_organizational_data_select [0..1]
  The event_context specifies the piece of product data the Event_reference refers to.
- offset : Duration [0..1] The offset specifies the amount of time before or after the defined event that shall be used to calculate the actual point in time.

#### 8.9.1.14 Class Interval_of_time

An Interval_of_time specifies a period of time.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- none

***Compositions***

- date_time_interval_assignment: date_time_interval_assignment[0..*]

***Associations***

- end_definition : Period_or_data_select[1]
  The end_definition defines the end of the Interval_of_time, either by referring to a bound, or specifying the extend of the Interval_of_time by a Duration. If the end_definition refers to an Event_reference or Date_time, this particular bound of the resulting interval is excluded from it.
- start_definition: Event_or_data_select[1]
  The start_definition defines the beginning of the Interval_of_time. The bound specified by the start_definition is included in the resulting interval.

#### 8.9.1.15 Class Organization

An Organization is a group of people involved in a particular business process.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- organization_name : String [1] The organization_name specifies the word or group of words used to refer to the Organization.
- organization_type : String [0..1]The organization_type specifies the type of the Organization. Where applicable the following values shall be used:
  - 'company': The organization_type specifies that the Organization is a company;
  - 'department': The organization_type specifies that the Organization is a department;
  - 'plant': The organization_type specifies that the Organization is a plant.
- id : String [1] The id specifies the identifier of the Organization.

***Compositions***

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Organization.

- person_organization_assignment : Person_organization_assignment [0..*]
  The Person_organization_assignment specifies the Person_organization_assignment that concerns this Organization
- date_and_person_organization : Date_and_person_organization [0..*]
  The Date_and_person_organization specifies the Date_and_person_organization which this Organization is part of.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Organization.

#### *Associations*

- postal_address : Address [0..1] The postal_address specifies the address where letter mail is delivered.
- delivery_address : Address [0..1]
  The delivery_address specifies the address where goods are delivered.
- visitor_address : Address [0..1]The visitor_address specifies the address where the organization receives visitors.

### 8.9.1.16 Class Organization_in_contract

An Organization_in_contract is a mechanism to associate the person who is signing a contract and the organization which the person is signing for, with a Contract.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role_of_organization: string[1] The role_of_organization specifies the function performed by the signing Organization with respect to the contract.

#### *Compositions*

- none

#### *Associations*

- contracted_organization: Organization[1]
  The contracted_organization specifies the organization that participates in the contract.
- signature : Date_and_person_organization[0..1]
  The signature specifies the set of Date_and_person_organization objects representing the personnel that signed the contract on behalf of the contracted organization and the date of the signature.

### 8.9.1.17 Class Person

A Person is an individual human being who has some relationship to product data. The Person shall always be identified in the context of one or more organizations.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- person_name : String [1] The person_name specifies the word or group of words used to refer to the Person.

#### *Compositions*

- person_in_organization : Person_in_organization [1..*]
  The Person_in_organization specifies the person_in_organization which this Person is assigned to.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Person.

#### *Associations*

- preferred_business_address : Address [0..1]
  The preferred_business_address specifies the location of the office of the Person.

### 8.9.1.18 Class Person_in_organization

A Person_in_organization is the specification of a Person in the context of an Organization.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1] The role specifies the relationship between the Person and the Organization.
- id : String [0..1] The id specifies an identifier of the person. The identifier shall be unique within the scope of the 'associated_organization'.

#### *Compositions*

- person_organization_assignment : Person_organization_assignment [0..*]
  The Person_organization_assignment specifies the Person_organization_assignment that concerns this Person_in_organization.
- date_and_person_organization : Date_and_person_organization [0..*]
  The Date_and_person_organization specifies the Date_and_person_organization which this Person_in_organization is part of.

#### *Associations*

- location : Address [0..1] The location specifies the relevant address of the Person_in_organization.
- associated_organization : Organization [1]
  The associated_organization specifies the Organization with which the Person is associated.

### 8.9.1.19 Class Person_organization_assignment

A Person_organization_assignment is an object that associates an Organization or a Person_in_organization with product data.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1]
  The role specifies the responsibility of the assigned Person or Organization with respect to the object that it is applied to. Where applicable the following values shall be used:
  - 'author': The referenced object has been created by the assigned Person or Organization. The author holds the copyright;
  - 'classification officer': The assigned Person or Organization is formally responsible for the classification of the referenced object;
  - 'creator': The referenced object has been created by the assigned Person or Organization;
  - 'custodian': The assigned Person or Organization is responsible for the existence and integrity of the referenced object;
  - 'customer': The assigned Person or Organization acts as a purchaser or consumer of the referenced object;
  - 'design supplier': The assigned Person or Organization is the one who delivers the data de-scribing the referenced object;
  - 'editor': The assigned Person or Organization is responsible for making any changes to any at-tribute of the referenced object;
  - 'id owner': The assigned Person or Organization is the one responsible for the designation of an identifier;
  - 'location': The assigned Organization is the place where the referenced object can be found or where it takes place;
  - 'manufacturer': The assigned Person or Organization is the one who produces the actual (physical) object;
  - 'owner': The assigned Person or Organization owns the referenced object, and has final say over its disposition and any changes to it;
  - 'supplier': The assigned Person or Organization is the one who delivers the actual (physical) object (e.g., a dealer);
  - 'wholesaler': The assigned Person or Organization is the one who is in the sales chain be-tween the manufacturer and the supplier.

#### *Compositions*

- description : String_select [0..1]
  The description specifies additional information about the Person_organization_assignment.

#### *Associations*

- is_applied_to : Date_time_person_organization_element_select [1..*]
  The is_applied_to specifies the object with which the Person_organization_assignment is associated.

#### 8.9.1.20 Class Security_classification

A Security_classification is the level of confidentiality that is required in order to protect product data against unauthorized usage.

***Base Class:***

- PLM_object (ABS)

***Attributes***

- none

***Compositions***

- name: String_select[1] The name specifies the word or group of words used to refer to the Security_classification.
- purpose: String_select[0..1] The purpose specifies the rationale behind the Security_classification.

***Associations***

- is_applied_to: Security_element_select[1]
  The is_applied_to specifies the objects with which the Security_classification is associated.

#### 8.9.1.21 Class Security_level

A Security_level is the specification of a level of security within some security classification scheme.

NOTE The values of Security_level are company specific.

***Base Class***

- PLM_root_object (ABS)

***Attributes***

- level_name: string[1] The level_name specifies the word or abbreviation used to refer to the Security_level.

***Compositions***

- security_classification: security_classification[0..*]

***Associations***

- used_classification_system : Classification_system[0..1]
  The used_classification_system specifies the Classification_system that contains the information about how to interpret the level of the Security_level.

### 8.9.2 Interfaces

#### 8.9.2.1 Interface Approval_element_select

This empty interface is realized by the following classes:

- Work_request

- Work_order
- Project
- Activity_method_assignment
- Activity_element
- Activity
- General_classification
- Classification_system
- Classification_association
- Specification_inclusion
- Specification_expression
- Specification_category
- Specification
- Product_structure_relationship
- Product_class
- Physical_instance_test_result
- Physical_instance
- Manufacturing_configuration (ABS)
- Design_constraint
- Configuration
- Complex_product (ABS)
- Class_structure_relationship
- Class_specification_association
- Class_inclusion_association
- Class_condition_association
- Class_category_association
- Document_version
- Document_representation (ABS)
- Document_file (ABS)
- Document
- Item_version
- Item_definition_relationship (ABS)
- Design_discipline_item_definition
- Physical_assembly_relationship
- Item_instance_relationship (ABS)

- Item_instance (ABS)
- Item_definition_instance_relationship (ABS)
- Assembly_substitute_relationship
- Process_plan
- Certification
- Contract
- Property_value_association (ABS)
- Simple_property_association
- Property (ABS)
- Material
- Geometric_model

#### 8.9.2.2 Interface Date_time_person_organization_element_select

This empty interface is realized by the following classes:

- Person_in_organization
- Event_reference
- Approval_status
- Work_request
- Work_order
- Project
- Activity_method_assignment
- Activity_element
- Activity
- General_classification
- Classification_system
- Classification_association
- Specification_inclusion
- Specification_expression
- Specification_category
- Specification
- Security_level
- Contract
- Product_structure_relationship
- Product_identification

- Product_class
- Physical_instance_test_result
- Physical_instance
- Manufacturing_configuration (ABS)
- Design_constraint
- Configuration
- Complex_product_relationship
- Complex_product (ABS)
- Class_structure_relationship
- Class_specification_association
- Class_inclusion_association
- Class_condition_association
- Class_category_association
- Document_version
- Document_representation (ABS)
- Document_file (ABS)
- Document
- Item_version_relationship
- Item_version
- Item_definition_relationship (ABS)
- Item
- Design_discipline_item_definition
- Physical_assembly_relationship
- Item_instance_relationship (ABS)
- Item_instance (ABS)
- Item_definition_instance_relationship (ABS)
- Process_plan
- Process_operation_resource_assignment
- Process_operation_occurrence
- Process_operation_definition
- Property_value_association (ABS)
- Simple_property_association
- Property (ABS)
- Material

- Assembly_substitute_relationship
- Geometric_model

#### 8.9.2.3 Interface Event_or_date_select

This empty interface is realized by the following classes:

- Event_reference
- Date_time

#### 8.9.2.4 Interface General_organizational_data_select

This empty interface is realized by the following classes:

- Person_in_organization
- Approval_status
- Work_request
- Work_order
- Project
- Activity_method_assignment
- Activity_element
- Activity
- General_classification
- Classification_system
- Classification_association
- Specification_inclusion
- Specification_expression
- Specification_category
- Specification
- Product_structure_relationship
- Product_identification
- Product_class
- Physical_instance_test_result
- Physical_instance
- Manufacturing_configuration (ABS)
- Design_constraint
- Configuration
- Complex_product_relationship
- Complex_product (ABS)

- Class_structure_relationship
- Class_specification_association
- Class_inclusion_association
- Class_condition_association
- Class_category_association
- Document_version
- Document_representation (ABS)
- Document_file (ABS)
- Document
- Item_version_relationship
- Item_version
- Item_definition_relationship (ABS)
- Item
- Design_discipline_item_definition
- Physical_assembly_relationship
- Item_instance_relationship (ABS)
- Item_instance (ABS)
- Item_definition_instance_relationship (ABS)
- Process_plan
- Process_operation_resource_assignment
- Process_operation_occurrence
- Process_operation_definition
- Property_value_association (ABS)
- Property (ABS)
- Material
- Geometric_model

#### 8.9.2.5 Interface Period_or_date_select

This empty interface is realized by the following classes:

- Event_reference
- Duration
- Date_time

#### 8.9.2.6 Interface Person_organization_select

***Compositions***

- date_and_person_organization : Date_and_person_organization [0..*]
- person_organization_assignment : Person_organization_assignment [0..*]

***Implemented By***

- Person_in_organization
- Organization

## 8.10 Package Configuration_management

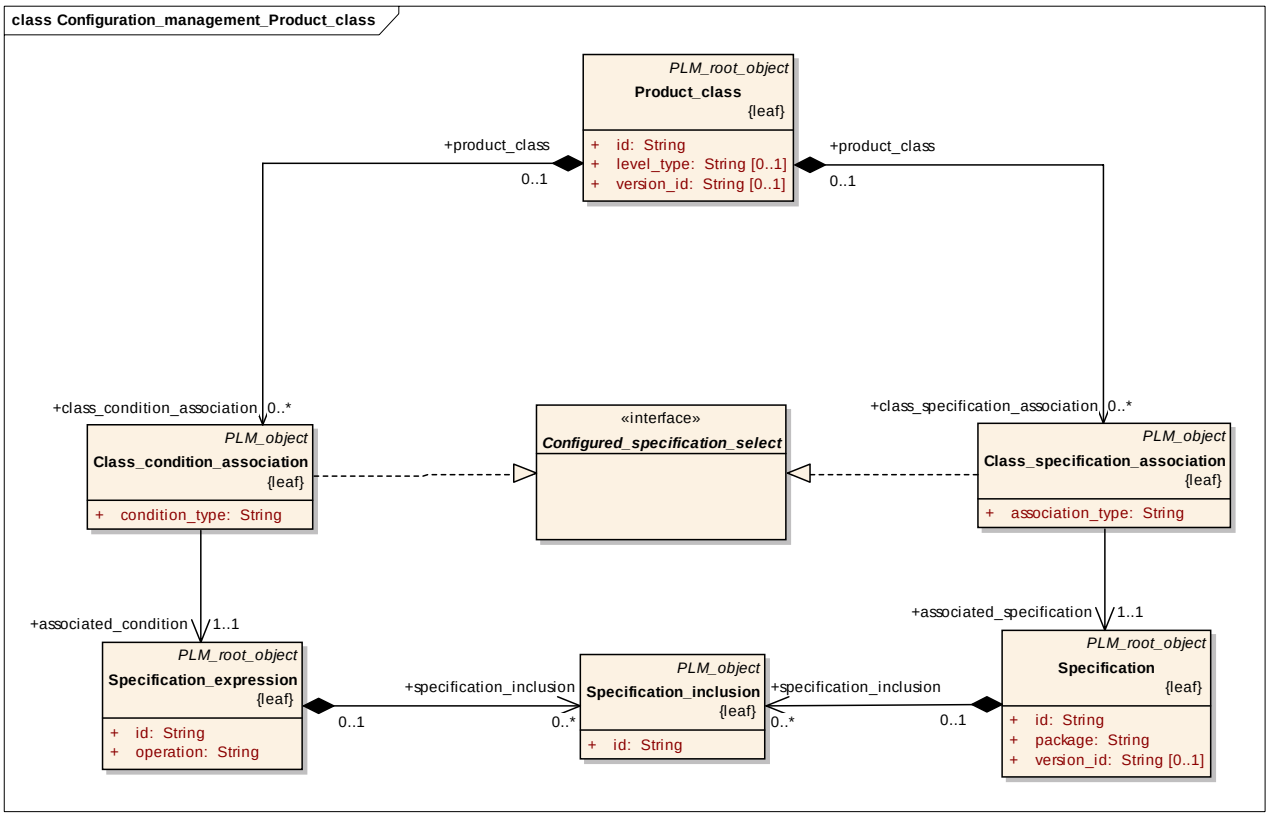


**Figure 8.35 - Configuration management - Product class condition and specification**

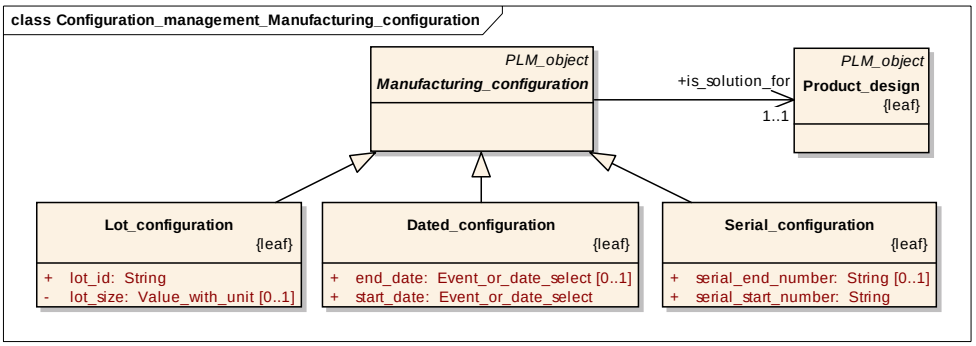


**Figure 8.36 - Configuration management - manufacturing configuration**

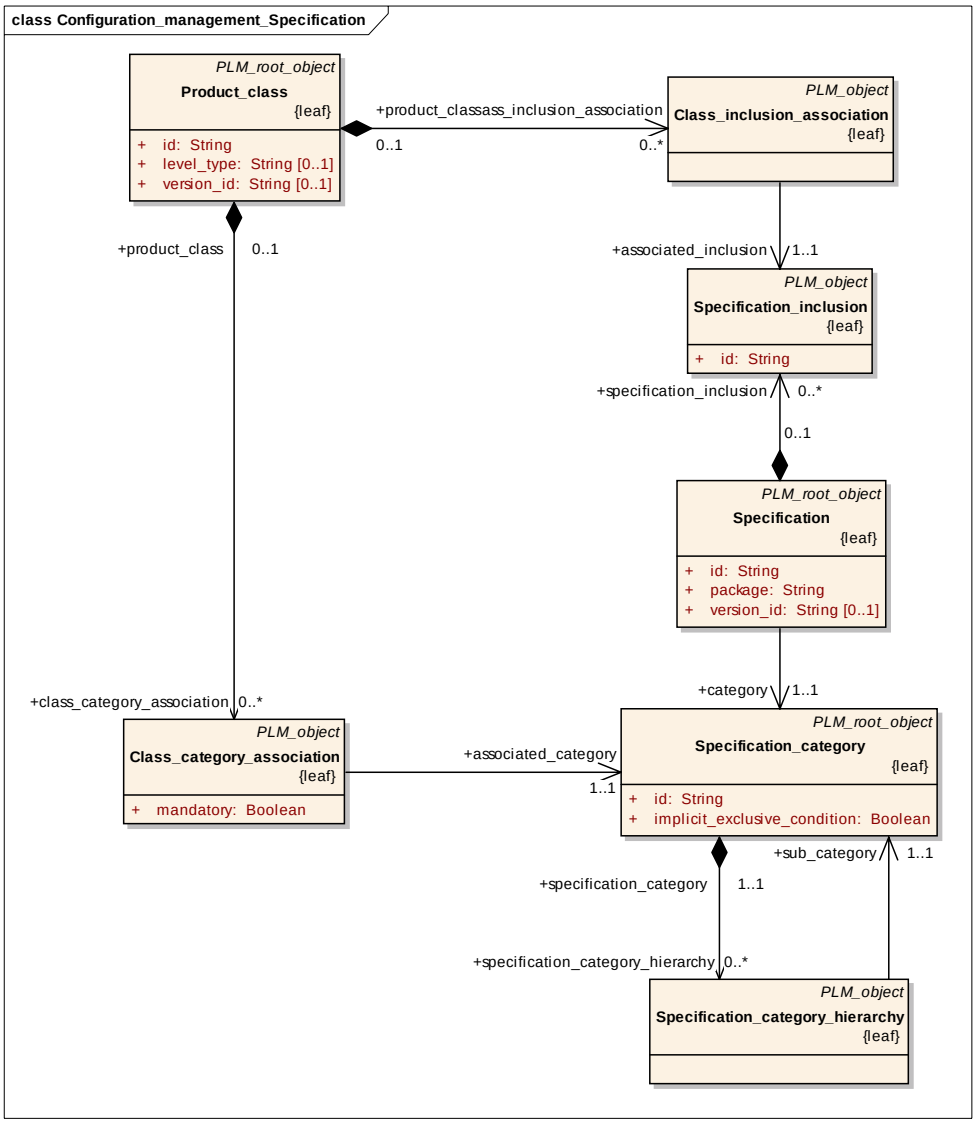


**Figure 8.37 - Configuration management - specification category and inclusion**

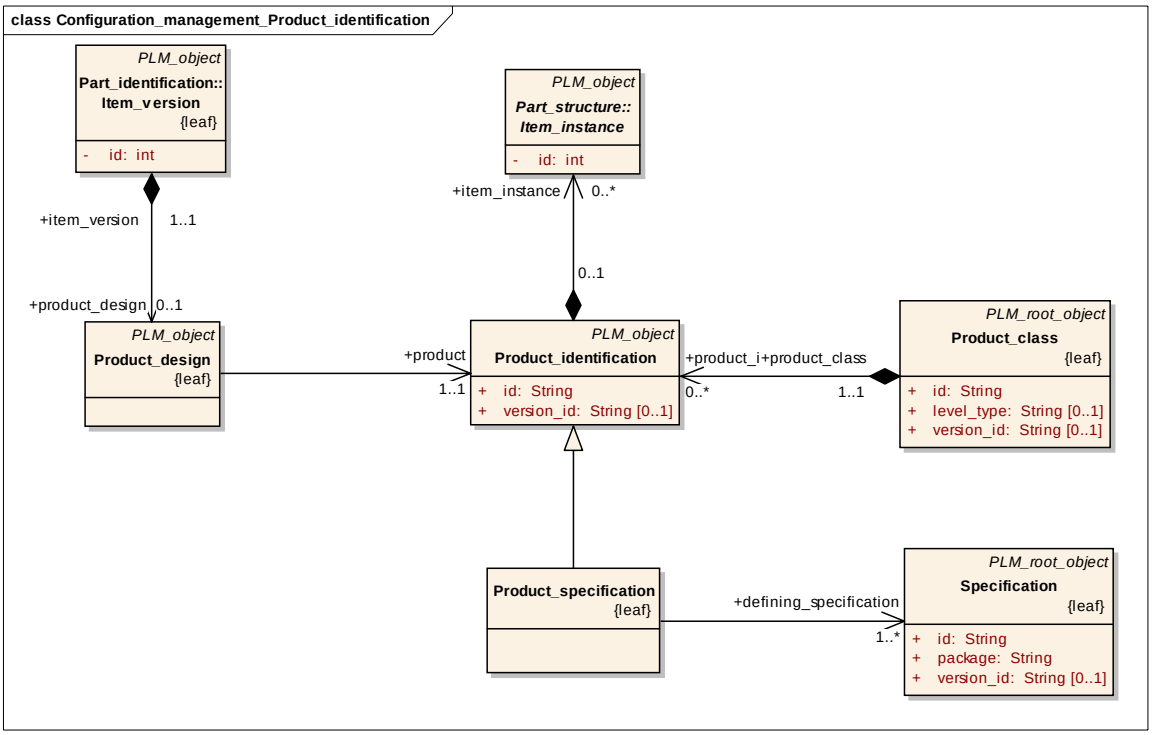


**Figure 8.38 - Configuration management - Product identification**

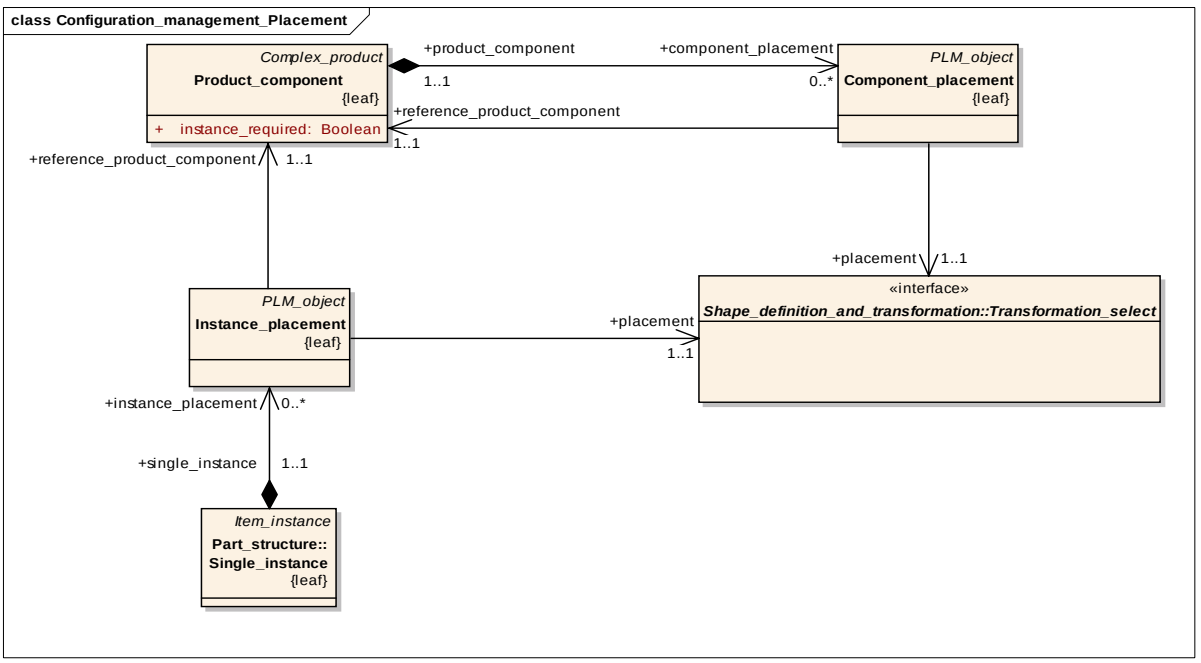


**Figure 8.39 - Configuration management - Component and instance placement**

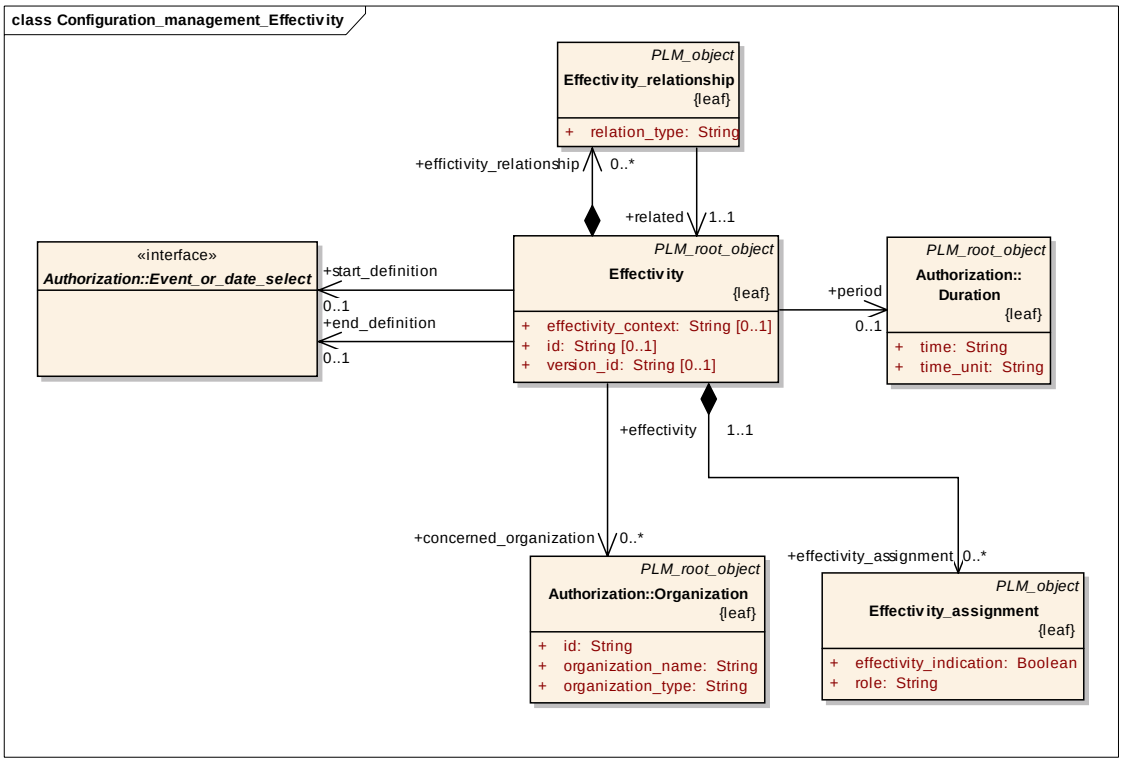


**Figure 8.40 - Configuration management - Effectivity**

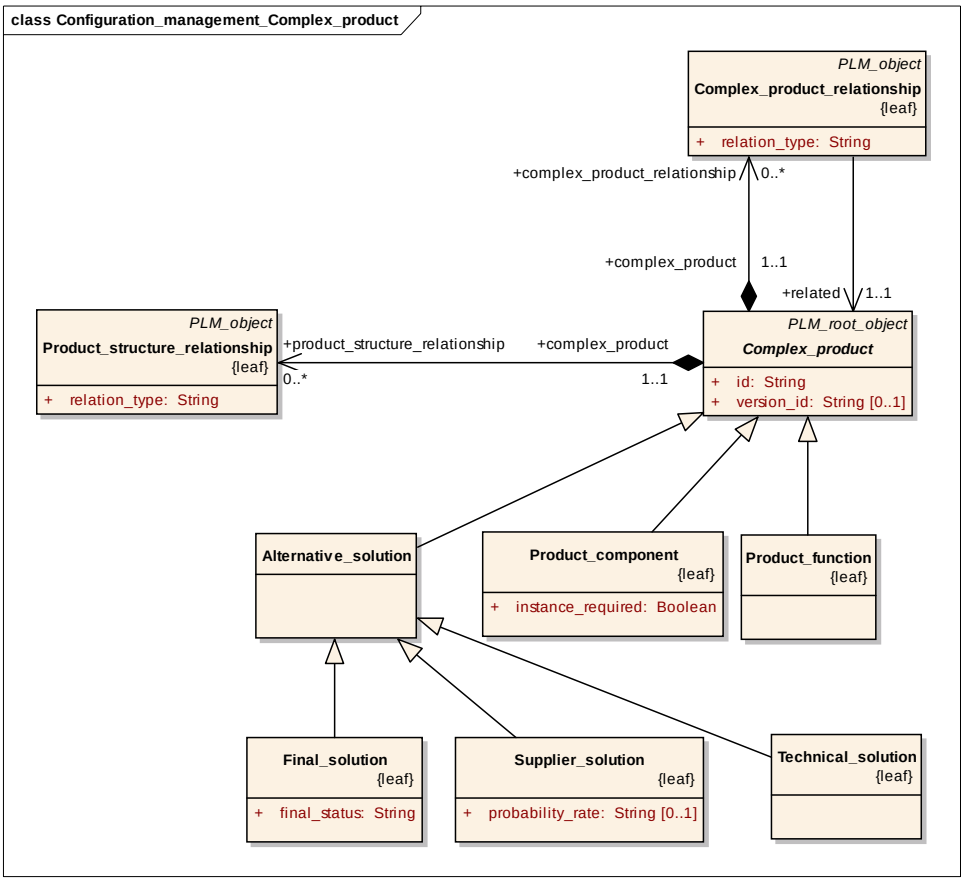


**Figure 8.41 - Configuration management - Complex product**

## 8.10.1 Classes

### 8.10.1.1 Class Alternative_solution

An Alternative_solution is the identification of one of potentially many mutually exclusive implementations of a Product_function or of a Product_component.

#### *Base Class*

- Complex_product (ABS)

#### *Attributes*

- none

#### *Compositions*

- configuration : Configuration [0..*]

  The configuration specifies the configuration that controls this Alternative_solution for its valid usage.

#### *Associations*

- base_element : Complex_product_select [1]
  The base_element specifies the object, for which the Alternative_solution provides a design alternative. All Alternative_solution objects for the same base_element are mutually exclusive.

### 8.10.1.2 Class Class_category_association

A Class_category_association is the association of a Specification_category with a Product_class. Additionally, this assignment specifies if the usage of one or more Specification objects belonging to this Specification_category, is mandatory or optional for all products of that Product_class.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- mandatory : Boolean [1] The mandatory specifies whether the Specification objects referring to the associated Specification_category have to be used or may be used (optional) for products within the referenced Product_class. A value of 'true' indicates that the usage is mandatory.

#### *Compositions*

- none

#### *Associations*

- associated_category : Specification_category [1]
  The associated_category specifies the Specification_category that is associated with the Product_class.

### 8.10.1.3 Class Class_condition_association

A Class_condition_association is the association of a Specification_expression with a Product_class.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- condition_type : String [1] The condition_type specifies the meaning of the association. Where applicable the following values shall be used:
  - 'design case': The Specification_expression specifies a condition when a given object has to be designed and verified. This value of the condition_type is for information only and shall not be interpreted when querying design cases or usage cases. For such a query, the value of the attribute 'configuration_type' of Configuration shall be evaluated;
  - 'identification': The Specification_expression specifies a condition that enables to distinguish the associated Product_class from other Product_class objects. This value is not applicable for a top level node in a hierarchy of Product_class objects. This identification is part of the identification of all sub classes of this Product_class;
  - 'part usage': The Specification_expression specifies a condition for the usage of the

components of an Alternative_solution, the usage of an Item_instance or for the application of a Process_plan or a Process_operation_occurrence in the products of the associated Product_class. In this case, the Class_condition_association shall be referenced by at least one Configuration object;
- 'validity': The Specification_expression specifies a condition that is used to verify a Product_specification for the associated Product_class. That means that the Specification_expression evaluates to 'true' if the set of Specification objects is valid; otherwise it evaluates to 'false' with the meaning that the specified object is invalid for the Product_class.
It is valid for all products belonging to the 'associated_product_class' in case of the condition types 'identification' and 'validity'.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Class_condition_association.

#### *Associations*

- associated_condition : Specification_expression [1]
  The associated_condition specifies the Specification_expression that is assigned to the Product_class.

### 8.10.1.4 Class Class_inclusion_association

A Class_inclusion_association is the assignment of a Specification_inclusion to a Prod-uct_class. This assignment contains the information that a particular Specification_inclusion applies for all products of that Product_class.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Class_inclusion_association.

#### *Association*

- associated_inclusion : Specification_inclusion [1]
  The associated_inclusion specifies the Specification_inclusion that is associated with the Product_class.

### 8.10.1.5 Class Class_specification_association

A Class_specification_association is an association of a Specification with a Product_class. This Specification serves as a potential characteristic of all products belonging to the Product_class.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- association_type : String [1] The association_type specifies the kind of availability of a particular Specification in a Product_class.

#### *Compositions*

- none

#### *Associations*

- associated_specification : Specification [1]
  The associated_specification specifies the Specification that is associated with the Product_class.

### 8.10.1.6 Class Class_structure_relationship

A Class_structure_relationship is an association between a Product_class object and either a Product_component or a Product_function object.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'functionality': The related Product_function is an element of the functional structure of the relating Product_class. This relation type shall only be used if the related object is a Product_function;
  - 'realization': The related Product_component fulfils, partially or fully, the requirements identified with the relating Product_class. This relation type shall only be used if the related object is a Product_component.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Class_structure_relationship.

#### *Associations*

- related : Product_function_component_select [1]
  The related specifies the Product_component or Product_function object related by the Class_structure_relationship.

### 8.10.1.7 Class Complex_product (ABS)

A Complex_product is an object with the capability that it can be realized by, decomposed into or specialized as Product_constituent objects in a functional, logical, or physical way.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Complex_product.
- version_id : String [0..1] The version_id identifies a version of the concept represented by a Complex_product.

#### *Compositions*

- product_structure_relationship : Product_structure_relationship [0..*]
  The product_structure_relationship specifies the product_structure_relationship where this Complex_product is decomposed functionally, logically, or physically into or realized by the related Product_constituent.
- design_constraint_association : Design_constraint_association [0..*]
  The design_constraint_association specifies the design_constraint_association so that the De-sign_constraint.affects this object.
- complex_product_relationship : Complex_product_relationship [0..*]
  The complex_product_relationship specifies the complex_product_relationship that relates the first of the two Complex_product objects.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Complex_product.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Complex_product.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- none

### 8.10.1.8 Class Complex_product_relationship

A Complex_product_relationship is a relationship between two Complex_product objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'derivation': the Complex_product_relationship defines a relationship where the related Complex_product is derived from the relating Complex_product;
  - 'replacement': The Complex_product_relationship defines a relationship where the related Complex_product is used in place of the relating Complex_product;
  - 'version hierarchy': the Complex_product_relationship defines a relationship where the related Complex_product is a sub version of the relating Complex_product;
  - 'version sequence': the Complex_product_relationship defines a relationship where the

relat-ing Complex_product is the preceding version and the related Complex_product is the following version.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Complex_product_relationship.

#### *Associations*

- related : Complex_product (ABS) [1]
  The related specifies the second of the two objects related by the Complex_product_relationship.

### 8.10.1.9 Class Component_placement

A Component_placement is the information pertaining to the placement of a Product_component, which is defined in its own Cartesian_coordinate_space, in the coordinate space of a reference Product_component.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- reference_product_component : Product_component [1]
  The reference_product_component specifies the high level Product_component that is defined in the reference coordinate space. A Model_property_association shall be assigned to the reference_product_component to define this reference coordinate space.
- placement : Transformation_select [1]
  The placement specifies the Geometric_model_relationship_with_transformation or the Template_instance that defines the position of the 'placed_component' relatively to the 'reference_product_component'. In the case of Template_instance, the scale shall be omitted or set to 1.0.

### 8.10.1.10 Class Configuration

A Configuration is the association of a Class_condition_association or a Class_specification_association object with a design or with a process in order to define a valid usage of it in the context of a certain Product_class.

#### *Base Class*

- PLM_object (ABS)

### *Attributes*

- configuration_type : String [1] The configuration_type specifies the valid usage of a Configuration object that is applied to the application object as configured_element. The following values shall be used:
  - 'design': The object referenced as 'configured_element' has to be designed and verified before it can actually be used in a given context. This context is specified by the Class_condition_association and Class_specification_association objects referenced as the 'is_solution_for'.
  - 'usage': The object referenced as the 'configured_element' is controlled by a Configuration. The Class_condition_association and Class_specification_association objects specify the us-age cases and are referenced as the 'is_solution_for'.
- inheritance_type : String [1] The inheritance_type specifies whether or not an inheritance scheme for the configuration in-formation in a hierarchical structure is applied to the application object referenced as the configured_element. The levels within such a hierarchy are defined through Product_structure_relationship objects or the attribute 'base_element' of Alternative_solution. The following values shall be used:
  - 'exception': No inheritance scheme is applicable and all required configuration information must be attached locally at the application object. The value indicates that the configuration in-formation may be inconsistent to the structural levels above it or that it is, on purpose, contradictory to it. Such a condition implies that an inheritance scheme shall not continue beyond this point in the product structure tree;
  - 'inherited': A scheme for inheritance of configuration information applies. The complete con-figuration information shall be collected from the different levels in the structure by evaluation of results. The results shall be evaluated using the logical AND to combine configuration informa-tion starting at the referenced configured_element and using the logical OR to combine alternatives. In addition, this evaluation shall consider related effectivity information. 'inherited' only applies for objects for which the same value of 'configuration_type' is defined;
  - 'local': No inheritance scheme is applicable and all required configuration information must be attached locally at the application object. Nevertheless, any potentially inherited configuration information of a higher level shall be consistent, i.e., be a subset of the locally defined configuration information.

### *Compositions*

- none

### *Associations*

- is_solution_for : Configured_specification_select [1]
  The is_solution_for specifies the characteristic or combination of characteristics for which the object referenced as the configured_element provides a solution or which is needed to control a process operation. These characteristics are defined by a Class_specification_association and combinations of characteristics are defined by a Class_condition_association where the attribute 'condition type' is 'part usage'.

#### 8.10.1.11 Class Dated_configuration

A Dated_configuration is a Manufacturing_configuration that applies onwards from a given date, or between a start and an end date.

#### *Base Class*

- Manufacturing_configuration (ABS)

#### *Attributes*

- start_date : Event_or_date_select [1]
  The start_date specifies the first date when the Dated_configuration is valid.
- end_date : Event_or_date_select [0..1]
  The end_date specifies the date and time when the validity of the 'configured_element' is not defined any longer.

#### *Compositions*

- none

#### *Associations*

- none

### 8.10.1.12 Class Descriptive_specification

A Descriptive_specification is a textual description of an object.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [0..1] The id specifies the identifier of the Descriptive_specification.

#### *Compositions*

- description : String_select [1] The description specifies the Descriptive_specification.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Descriptive_specification.

#### *Associations*

- none

### 8.10.1.13 Class Design_constraint

A Design_constraint is a requirement that has to be considered in the design process of a Complex_product. This constraint may be geometry based.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- constraint_id : String [1] The constraint_id specifies the identifier of the Design_constraint.

#### *Compositions*

- design_constraint_relationship : Design_constraint_relationship [0..*]
  The design_constraint_relationship specifies the design_constraint_relationship that relates the first of the two Design_constraint objects.
- description : String_select [0..1]The description specifies additional information about the Design_constraint.
- name : String_select [0..1] The name specifies the word or group of words by which the Design_constraint is referred to.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Design_constraint.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- is_valid_for : Product_class [0..*]The is_valid_for specifies the set of Product_class objects that are affected by the Design_constraint.

### 8.10.1.14 Class Design_constraint_association

A Design_constraint_association is a mechanism to associate a Design_constraint with an object that is subject to the constraint indicated.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- name : String_select [0..1] The name specifies the word or group of words by which the Design_constraint_association is referred to.

#### *Associations*

- is_based_on : Design_constraint [1]The is_based_on specifies the Design_constraint that represents the constraint.

### 8.10.1.15 Class Design_constraint_relationship

A Design_constraint_relationship is a relationship between two Design_constraint objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Design_constraint_relationship.

#### *Associations*

- related : Design_constraint [1] The related specifies the second of the two Design_constraint objects related by the Design_constraint_relationship.

### 8.10.1.16 Class Design_constraint_version

A Design_constraint_version is a particular version of a Design_constraint.

#### *Base Class*

- Design_constraint

#### *Attributes*

- version_id : String [1] The version_id specifies the identification of a particular version of a Design_constraint. The version_id shall be unique within the scope of a Design_constraint.

#### *Compositions*

- none

#### *Associations*

- none

### 8.10.1.17 Class Effectivity

An Effectivity is the identification of the valid use of an aspect of product data tracked by date or event.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [0..1] The id specifies the identifier of the Effectivity.
- version_id : String [0..1] The version_id specifies the identification of a particular version of the Effectivity.
- effectivity_context : String [0..1]The effectivity_context specifies the life cycle stage for which the Effectivity is valid.

#### *Compositions*

- effectivity_assignment : Effectivity_assignment [0..*] The effectivity_assignment specifies the effectivity_assignment which this Effectivity is assigned to.
- description : String_select [0..1]The description specifies additional information about the Effectivity.

#### *Associations*

- end_definition : Event_or_date_select [0..1]
  The end_definition specifies the end of the period. The bound specified by the end_definition is excluded from the interval of effectivity.
- start_definition : Event_or_date_select [0..1]
  The start_definition specifies the start of the period. The bound specified by the start_definition is included in the interval of effectivity.
- period : Duration [0..1] The period specifies the period of time in which the Effectivity is defined, either starting at the point in time specified by 'start_definition' or ending at the point in time specified by 'end_definition'. period shall be specified with a positive value.
- concerned_organization : Organization [0..*]
  The concerned_organization specifies the set of Organization objects in which the Effectivity is valid.

### 8.10.1.18 Class Effectivity_assignment

An Effectivity_assignment associates an Effectivity with the object whose effectivity is controlled by the associated Effectivity. The association of an Effectivity to product data does not imply any statement concerning the effectivity outside of the specified interval. The same applies in the absence of any assigned effectivity, i.e. no statement concerning the effectivity is implied.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1] The role specifies the relationship between the Effectivity and the object that has an effectivity assigned to it. Where applicable the following values shall be used:
  - 'actual': The actual period during which the Effectivity lasted;
  - 'planned': The period associated with the Effectivity defines a planned period of time during which the associated object is or was supposed to be effective;
  - 'required': The associated object must be kept effective for this period.
- effectivity_indication : Boolean [1]
  The effectivity_indication specifies whether the assigned_effectivity defines a period of effectivity (value equal 'TRUE') or a period of ineffectivity (value equal 'FALSE') for the effective_element. In the first case, use of the effective_element is or was valid during the considered period.

#### *Compositions*

- none

#### *Associations*

- effective_element : Effective_element_select [1..*]
  The effective_elements specify the objects that have an Effectivity assigned to it.

### 8.10.1.19 Class Effectivity_relationship

An Effectivity_relationship is a relationship between two Effectivity objects.

**Note –** Sometimes the effectivity is not dependent on particular dates but on the effectivity of other items. In this case the dates are not instantiated and there is an Effectivity_relationship to the reference Effectivity.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type: string[1] The relation_type specifies the meaning of the relationship.

#### *Compositions*

- description: String_select[0..1]
  The description specifies additional information about the Effectivity_relationship.

#### *Associations*

- related: Effectivity[1] The related specifies the second of the two Effectivity objects related by the Effectivity_relationship. NOTE The semantics of this attribute are defined by the attribute 'relation_type'.

### 8.10.1.20 Class Final_solution

A Final_solution is the specification of a set of additional sensual characteristics that can be applied to an Item_instance that represents a neutral part in order to finalize its definition.

#### *Base Class*

- Alternative_solution

#### *Attributes*

- final_status : String [1] The final_status specifies the level of completion between the neutral part and the final part.

#### *Compositions*

- none

#### *Associations*

- final_specification : Final_definition_select [1..*]
  The final_specification specifies the means of finalization that is applied to the neutral part and which may be objects of type Descriptive_specification, Physical_instance, or Design_discipline_item_definition.

### 8.10.1.21 Class Instance_placement

An Instance_placement is the information pertaining to the placement of a Single_instance, which is defined in its own Cartesian_coordinate_space, in the coordinate space of a reference Product_component.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- reference_product_component : Product_component [1]
  The reference_product_component specifies the Product_component that specifies indirectly the reference coordinate space. A Model_property_association shall be assigned to the reference_product_component to define this reference coordinate space.
- placement : Transformation_select [1]
  The placement specifies the Geometric_model_relationship_with_transformation or the Template_instance that defines the position of the 'placed_instance' relatively to the 'reference_product_component'. In the case of Template_instance, the scale shall be omitted or set to 1.0.

### 8.10.1.22 Class Item_function_association

An Item_function_association is a mechanism to relate a Product_function and a Design_discipline_item_definition.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- association_type : String [1] The association_type specifies the kind of association.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Item_function_association.

#### *Associations*

- associated_function : Product_function [1]
  The associated_function specifies the associated Product_function.

### 8.10.1.23 Class Lot_configuration

A Lot_configuration is a Manufacturing_configuration that applies to a given production batch of the product that is related with the object referred to as 'is_solution_for'.

#### *Base Class*

- Manufacturing_configuration (ABS)

#### *Attributes*

- lot_id : String [1] The lot_id specifies the identification of the batch for which the Lot_configuration applies.
- lot_size : Value_with_unit (ABS) [1]
  The lot_size specifies the size of the batch for which the Lot_configuration applies.

#### *Compositions*

- none

#### *Associations*

- none

### 8.10.1.24 Class Manufacturing_configuration (ABS)

A Manufacturing_configuration is the association of a Product_design with an Item_instance.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- concerned_organization : Organization [0..*]
  The concerned_organization specifies the Organization in which the Manufacturing_configuration is valid. The case where the concerned_organization is an empty set means that the Manufacturing_configuration regards any organization that may consider the 'configured_element'.
- is_solution_for : Product_design [1]The is_solution_for specifies the design for which an Item_instance is configured.

### 8.10.1.25 Class Physical_instance

A Physical_instance is the denomination of a physically realized object. A Physical_instance may be identified by a serial number. A lot id may be provided additionally to the serial number.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- serial_number : String [0..1] The serial_number is an identifier that distinguishes one Physical_instance from another.
- lot_id : String [0..1] The lot_id specifies the identifier of the lot the Physical_instance is part of.

- inventory_number : String [0..1]The inventory_number specifies an alphanumerical string to identify an item in the detailed list of articles, such as goods and chattels, found in the possession of a person or enterprise.

#### *Compositions*

- physical_instance_test_result : Physical_instance_test_result [0..*]
  The physical_instance_test_result specifies the physical_instance_test_result for which this Physical_instance was the subject of the test activity.
- description : String_select [0..1]The description specifies additional information about the Physical_instance.
- physical_assembly_relationship : Physical_assembly_relationship [0..*]
  The physical_assembly_relationship specifies the physical_assembly_relationship for which this Physical_instance serves as the assembly in the physical structure.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Physical_instance.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Physical_instance.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- is_realization_of : Physical_instance_definition_select [0..1]
  The is_realization_of specifies the Product_identification or the Design_discipline_item_definition that collects the information defining the Physical_instance.

### 8.10.1.26 Class Physical_instance_test_result

A Physical_instance_test_result is a mechanism to associate a Physical_instance with measurements made on this Physical_instance.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Physical_instance_test_result.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Physical_instance_test_result.
- Document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Physical_instance_test_result.

#### *Associations*

- test_result : Property_value_representation [0..*]
  The test_result specifies the characteristics that were determined by the performed test.
- test_activity : Test_activity_select [0..1]
  The test_activity specifies the Activity or the Process_operation_occurrence that has lead to the test result.

### 8.10.1.27 Class Product_class

A Product_class is the identification of a set of similar products to be offered to the market. Product_class objects that are related to each other by a Product_class_relationship do not inherit any characteristics from each other.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Product_class that shall be unique.
- level_type : String [0..1] The level_type specifies the level or category of this Product_class in a hierarchical structure of Product_class objects. The level_type shall only be used if and only if the level_type is specified in the the context of the unit of functionality 'specification_control' (UoF S7).
- version_id : String [0..1] The version_id specifies the identification of a particular version of a Product_class.

#### *Compositions*

- product_identification : Product_identification [0..*]
  The product_identification specifies the product_identification of the product that belongs to this Product_class.
- description : String_select [0..1]The description specifies additional information about the Product_class.
- name : String_select [0..1] The name specifies the word or group of words by which the Product_class is referred to.
- class_structure_relationship : Class_structure_relationship [0..*]
  The class_structure_relationship specifies the class_structure_relationship that relates this Product_class.
- class_specification_association : Class_specification_association [0..*]
  The class_specification_association specifies the class_specification_association that is valid for this Product_class.
- class_inclusion_association : Class_inclusion_association [0..*]
  The class_inclusion_association specifies the class_inclusion_association that is valid for this Product_class.
- class_condition_association : Class_condition_association [0..*]
  The class_condition_association specifies the class_condition_association that is valid for this Product_class.
- class_category_association : Class_category_association [0..*]
  The class_category_association specifies the class_category_association that is valid for this Product_class.

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Product_class.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Product_class.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- none

### 8.10.1.28 Class Product_class_relationship

A Product_class_relationship is a relationship between two Product_class objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : string[1] The relation_type specifies the meaning of the relationship. NOTE The relationship does not imply inheritance of any kind between the application objects that are related.

#### *Compositions*

- description : String_select[0..1]
  The description specifies additional information about the Product_class_relationship.

#### *Associations*

- related : Product_class[1] The related specifies the second of the two Product_class objects related by the Product_class_relationship. NOTE The semantics of this attribute are defined by the attribute relation_type.

### 8.10.1.29 Class Product_component

A Product_component is an element in a conceptual product structure.

#### *Base Class*

- Complex_product (ABS)

#### *Attributes*

- instance_required : Boolean [1]The instance_required specifies if the existence of a corresponding Item_instance is required for the various Alternative_solution objects of that Product_component. A value of 'true' indicates that a corresponding Item_instance is required.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Product_component.

- name : String_select [0..1] The name specifies the word or group of words by which the Product_component is referred to.
- configuration : Configuration [0..*]
  The configuration specifies the configuration that controls this Product_component for its valid usage.
- component_placement : Component_placement [0..*]
  The component_placement specifies the component_placement that is positioned with respect to this Product_component.

#### *Associations*

- is_relevant_for : Application_context [0..*]
  The is_relevant_for specifies the Application_context objects in which the Product_component has to be considered.
- is_influenced_by : Class_category_association [0..*]
  The is_influenced_by specifies the Specification_category objects that impact the design of a solution for the Product_component in the context of the Product_class objects that are referred to by the Class_category_association objects.

### 8.10.1.30 Class Product_design

A Product_design is a mechanism to associate an Item_version with its corresponding Product_identification.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- product : Product_identification [1]
  The product specifies the Product_identification that represents the requirements.

### 8.10.1.31 Class Product_function

A Product_function is a behavior or an action expected from a product.

#### *Base Class*

- Complex_product (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Product_function.
- name : String_select [0..1] The name specifies the word or group of words by which the Product_function is referred to.
- configuration : Configuration [0..*]
  The configuration specifies the configuration that controls this Product_function for its valid usage.

#### *Associations*

- is_relevant_for : Application_context [0..*]
  The is_relevant_for specifies the Application_context objects in which the Product_function has to be considered.

### 8.10.1.32 Class Product_identification

A Product_identification identifies a manufacturable object, or expected as so. A Product_identification is defined with respect to the Product_class it is a member of.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- version_id : String [0..1] The version_id specifies the identification of a particular version of a Product_identification.
- id : String [1] The id specifies the identifier of the Product_identification.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Product_identification.
- name : String_select [0..1] The name specifies the word or group of words by which the Product_identification is referred to.
- item_instance : Item_instance (ABS) [0..*]
  The item_instance specifies the item_instance for which this Product_identification serves as a definition.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Product_identification.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- none

### 8.10.1.33 Class Product_specification

A Product_specification is a Product_identification for which one or more additional Specification objects enhance the characterization provided for the associated Product_class.

#### *Base Class*

- Product_identification

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- defining_specification : Specification [1..*]
  The defining_specification specifies the set of Specification objects necessary to discriminate the Product_specification within its Product_class.

### 8.10.1.34 Class Product_structure_relationship

A Product_structure_relationship is an association between a Complex_product and a Product_constituent, in which the Product_constituent is a functional, logical, or physical component or a realization of the Complex_product.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1]
  The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'decomposition': The related Product_constituent is one of potentially more components of the relating Complex_product. This relation type shall only be used for Complex_product and Product_constituent of the same type;
  - 'functionality': The related Product_constituent is an element of the functional structure of the relating Complex_product. This relation type shall only be used with a Complex_product of type Alternative_solution or Product_component and with a Product_constituent of type Product_function;
  - 'occurrence': The related Product_constituent is an occurrence defined by the relating Complex_product. This relation type shall only be used if related Product_constituent is of type Product_component;
  - 'realization': The related Product_constituent is a means for fulfilling, either partially or fully, the requirements identified with the relating Complex_product. This relation type shall be used only when the Complex_product and the Product_constituent are of different types;
  - 'specialization': The related Product_constituent fulfils the requirements of the relating Complex_product in a more specific way than defined for the relating Complex_product. This relation type shall only be used for Product_constituent and Complex_product of the same type.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Product_structure_relationship.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Product_structure_relationship.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- related : Product_constituent_select [1]The related specifies the Product_constituent that is a functional, logical, or physical component or a realization of the relating Complex_product.

### 8.10.1.35 Class Retention_period

A Retention_period is the definition of a period of time that product data needs to be maintained due to organizational policy or legal requirements.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- retention_purpose : String_select[0..1]
  The retention_purpose specifies the rationale behind the Retention_period.

#### *Associations*

- start_definition : Event_or_date_select[1]
  The start_definition specifies the point in time at which the Retention_period starts.
- earliest_end_definition : Period_or_date_select[[1]
  The earliest_end_definition specifies the point in time after which any object the Retention_period is applied to may be deleted or destroyed. In this context deletion or destruction applies to all subordinate objects that are not referenced by other objects.
- is_applied_to: General_organizational_data_select[1]
  The is_applied_to specifies the objects whose existence is controlled by the Retention_period. NOTE The master document is the one for which earliest_end_definition and latest_end_definition are the same.
- latest_end_definition: Period_or_date_select[1]
  The latest_end_definition specifies the point in time before which any objects that the Retention_period is applied to shall be deleted or destroyed. In this context deletion or destruction applies to all subordinate objects that are not used by other objects.

### 8.10.1.36 Class Serial_configuration

A Serial_configuration is a Manufacturing_configuration that applies onwards from a given serial number of the product that is considered within the object referred to as 'is_solution_for'.

#### *Base Class*

- Manufacturing_configuration (ABS)

#### *Attributes*

- serial_start_number : String [1]The serial_start_number specifies the serial number of that instance of the product that is the first instance for which the Serial_configuration applies.
- serial_end_number : String [0..1]The serial_end_number specifies the serial number of that instance of the product that is the last instance for which the Serial_configuration applies.

#### *Compositions*

- none

#### *Associations*

- none

### 8.10.1.37 Class Specification

A Specification is a characteristic of a product. A Specification discriminates one product from other members of the same Product_class. A Specification refers to a Specification_category that completes the semantics of the Specification.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Specification that shall be unique within the scope of a Specification_category.
- version_id : String [0..1] The version_id specifies the identification of a particular version of a Specification.
- package : Boolean [1] The package specifies whether this Specification represents a package of Specification objects or not. Such a Specification combines those Specification objects that shall be offered to the market as a set. In the case where package is 'true', there shall be exactly one Specification_inclusion per Product_class considered, that refers to this Specification as 'if_condition'. The Specification objects that are members of the package, shall be specified as included_specification.

#### *Compositions*

- specification_inclusion : Specification_inclusion [0..*]
  The specification_inclusion specifies the specification_inclusion for which this Specification serves as the condition for the inclusion.
- description : String_select [0..1]The description specifies additional information about the Specification.
- name : String_select [0..1] The name specifies the word or group of words by which the Specification is referred to.

- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Specification.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Specification.

#### *Associations*

- category : Specification_category [1]
  The category specifies the Specification_category that completes the semantics of the Specification.

### 8.10.1.38 Class Specification_category

A Specification_category is the definition of a set of Specification objects serving the same purpose.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- implicit_exclusive_condition : Boolean [1]
  The implicit_exclusive_condition specifies whether the Specification objects within the Specification_category are mutually exclusive for the production of one particular product. A value of 'true' indicates that the referenced objects are mutually exclusive for the production of the particular product.
- id : String [1] The id specifies the identifier of the Specification_category that shall be unique.

#### *Compositions*

- specification_category_hierarchy : Specification_category_hierarchy [0..*]
  The specification_category_hierarchy specifies the specification_category_hierarchy for which this Specification_category is the higher level.
- description : String_select [1] The description specifies information about the Specification_category.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Specification_category.
- alias_identification : Alias_identification [0..*]
  The Alias_identification specifies the Alias_identification that is applied to this Specification_category.

#### *Associations*

- none

### 8.10.1.39 Class Specification_category_hierarchy

A Specification_category_hierarchy is used to build up hierarchical structures of Specification_category objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- sub_category : Specification_category [1]
  The sub_category is the lower level of Specification_category in Specification_category_hierarchy.

### 8.10.1.40 Class Specification_expression

A Specification_expression is a combination of Specification objects formed by Boolean operations.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- operation : String [1]
  The operation specifies the kind of Boolean operation. Four kinds of operations are permitted:
  - 'and': All of the identified Specification objects shall be used;
  - 'or': A subset or all of the identified Specification objects shall be used;
  - 'oneof': Exactly one of the identified Specification objects shall be used;
  - 'not': The identified Specification shall not be used.
- id : String [0..1]
  The id specifies the identifier of the Specification_expression.

#### *Compositions*

- specification_inclusion : Specification_inclusion [0..*]
  The specification_inclusion specifies the specification_inclusion for which this Specification_expression serves as the condition for the inclusion.
- description : String_select [0..1]
  The description specifies additional information about the Specification_expression.

#### *Associations*

- operand : Specification_operand_select [1..*]
  The operand specifies the operands of the Boolean operation that are either Specification objects or other Specification_expression objects.

### 8.10.1.41 Class Specification_inclusion

A Specification_inclusion is the representation of the statement that specifies that the application of a Specification or of a Specification_expression implies the inclusion of an additional Specification or Specification_expression.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- id : String [0..1] The id specifies the identifier of the Specification_inclusion.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Specification_inclusion.

#### *Associations*

- included_specification : Specification_operand_select [1]
  The included_specification specifies the Specification or the Specification_expression objects that are to be included. The included_specification shall not reference a Specification_expression with an operation of type 'or' or 'oneof', except for negating expressions, i.e., as participants in an expression preceded by a 'not' operator. Expressions of operator 'not' shall not be nested within each other.

### 8.10.1.42 Class Supplier_solution

A Supplier_solution is an alternative solution provided by a particular supplier.

#### *Base Class*

- Alternative_solution

#### *Attributes*

- probability_rate : String [0..1] The probability_rate specifies the share that is assigned to the supplier in the context of the base element.

#### *Compositions*

- none

#### *Associations*

- supplier : Organization [1] The supplier specifies the Organization that acts as supplier for the Supplier_solution.

### 8.10.1.43 Class Technical_solution

A Technical_solution is an alternative solution where the functional requirements are fulfilled in a certain technical way.

#### *Base Class*

- Alternative_solution

#### *Attributes*

- none

#### *Compositions*

- description : String_select [1]  The description specifies additional information about the Technical_solution.

#### *Associations*

- none

### 8.10.2 Interfaces

#### 8.10.2.1 Interface Complex_product_select

This empty interface is realized by the following classes:

- Product_function
- Product_component
- Alternative_solution

#### 8.10.2.2 Interface Configured_specification_select

This empty interface is realized by the following classes:

- Class_specification_association
- Class_condition_association

#### 8.10.2.3 Interface Configured_item_select

#### *Compositions*

- configuration : Configuration [0..*]

#### *Implemented By*

- Process_operation_occurrence
- Product_function
- Product_component
- Alternative_solution
- Process_plan
- Item_instance

#### 8.10.2.4 Interface Effective_element_select

This empty interface is realized by the following classes:

- Classification_system
- Specification_inclusion
- Specification_expression
- Specification_category

- Specification
- Product_structure_relationship
- Product_identification
- Product_class
- Design_constraint
- Configuration
- Security_classification
- Complex_product_relationship
- Complex_product (ABS)
- Class_structure_relationship
- Class_specification_association
- Class_inclusion_association
- Class_condition_association
- Class_category_association
- Document_version
- Document_representation (ABS)
- Document_file (ABS)
- Document
- Item_version
- Item_definition_relationship (ABS)
- Item
- Item_instance_relationship (ABS)
- Item_instance (ABS)
- Item_definition_instance_relationship (ABS)
- Assembly_substitute_relationship
- Process_plan
- Process_operation_resource_assignment
- Process_operation_occurrence_relationship
- Process_operation_occurrence
- Process_operation_definition_relationship
- Process_operation_definition
- Property_value_association (ABS)
- Simple_property_association
- Property (ABS)

- Material
- Geometric_model

#### 8.10.2.5 Interface Final_definition_select

This empty interface is realized by the following classes:

- Physical_instance
- Descriptive_specification
- Design_discipline_item_definition

#### 8.10.2.6 Interface Instance_definition_select

***Compositions***

- item_instance : Item_instance [0..*]

***Implemented By***

- Product_identification
- Design_discipline_item_definition

#### 8.10.2.7 Interface Physical_instance_definition_select

This empty interface is realized by the following classes:

- Product_identification
- Design_discipline_item_definition

#### 8.10.2.8 Interface Product_function_component_select

This empty interface is realized by the following classes:

- Product_function
- Product_component

#### 8.10.2.9 Interface Specification_operand_select

***Compositions***

- specification_inclusion : Specification_inclusion [0..*]

***Implemented By***

- Specification
- Specification_expression

#### 8.10.2.10 Interface Test_activity_select

This empty interface is realized by the following classes:

- Activity
- Process_operation_occurrence

## 8.11 Package Change_and_work_management

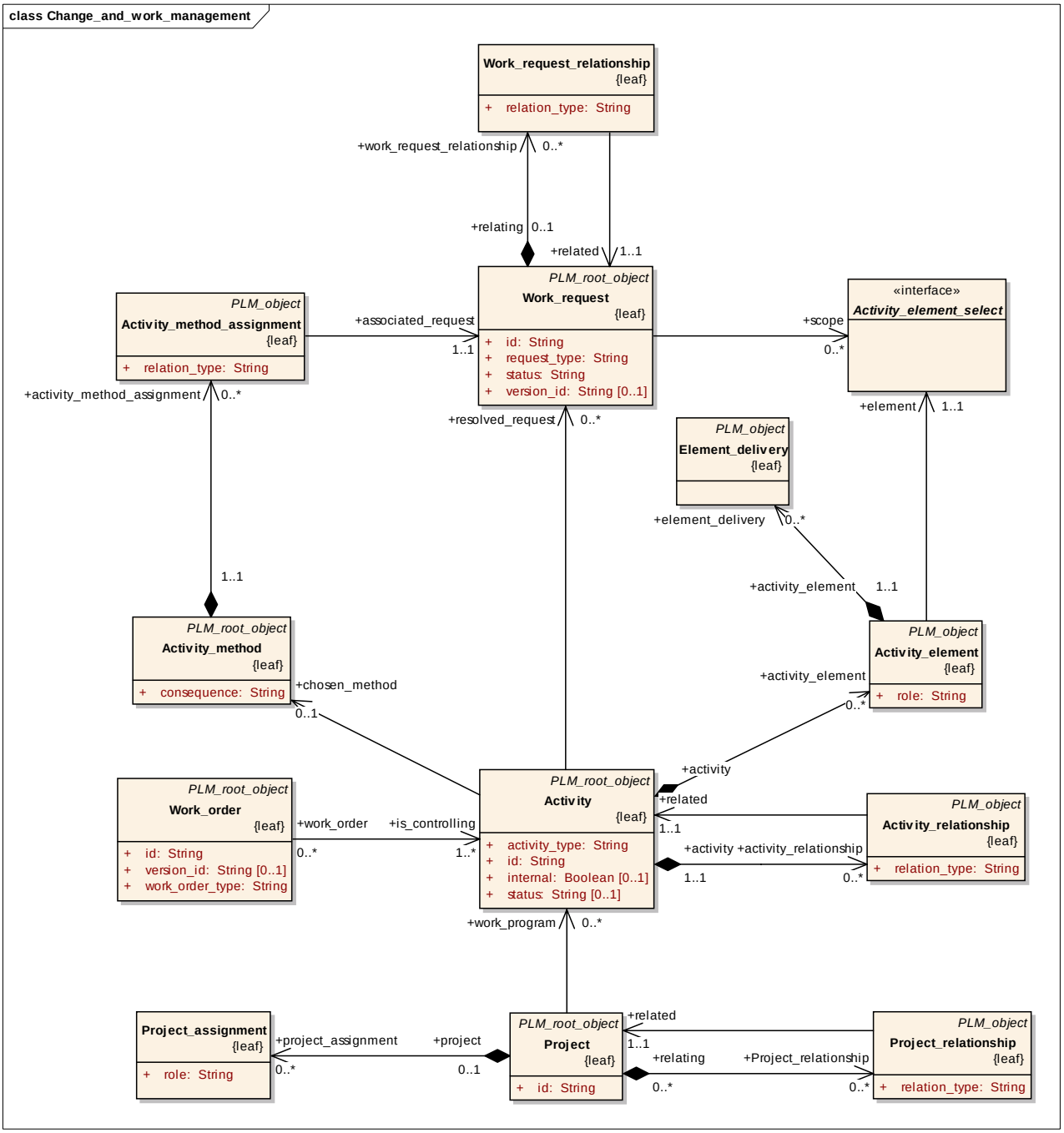


**Figure 8.42 - Change management**

## 8.11.1 Classes

### 8.11.1.1 Class Activity

An Activity is the fact of achieving or accomplishing an action.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- activity_type : String [1] The activity_type specifies the purpose of the Activity. Where applicable the following values shall be used:
  - 'amendment': An Activity to add information to product data;
  - 'analysis': An Activity to determine the behaviour of an element under certain physical circumstances;
  - 'cancellation': An Activity to delete an element from the bill of material or to cancel the whole bill of material;
  - 'delivery change': An Activity to change the delivery schedule of an element;
  - 'design change': An Activity to change the design of an item or an assembly; this might include changes to the geometry or to properties of the object;
  - 'design': An Activity concerning the development of a design of an item;
  - 'mock-up creation': An Activity to create an experimental model or replica of an item;
  - 'prototype building': An Activity to manufacture a preliminary version of an item;
  - 'rectification': An Activity to correct the data, documentation or structure associated with an item;
  - 'restructuring': An Activity to create a new structure or position within a bill of material without changing the data associated with the items in the bill of material;
  - 'spare part creation': An Activity to design a spare part or to classify an item as a spare part;
  - 'stop notice': An Activity to stop the manufacturing process of an item;
  - 'testing': An Activity to test an item;
  - 'work definition': An Activity to manage several sub-activities related to this Activity by an Activity_relationship with a 'relation_type' of value 'decomposition'.
- id : String [1] The id specifies the identifier of the Activity.
- status : String [0..1] The status specifies the level of completion of the Activity.
- internal : Boolean [0..1] The internal specifies whether the activity is carried out within the organization that initiated the activity. A value of 'true' indicates that the activity is carried out within this particular organization.

#### *Compositions*

- activity_relationship : Activity_relationship [0..*]
  The Activity_relationship specifies the Activity_relationship that relates the first of the two Activity objects.
- activity_element : Activity_element [0..*]
  The Activity_element specifies the Activity_element that belongs to this Activity.
- description : String_select [0..1]The description specifies additional information about the Activity.

- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Activity.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- chosen_method : Activity_method [0..1]
  The chosen_method specifies the Activity_method used to carry out the Activity.
- actual_start_date : Date_time [0..1]
  The actual_start_date specifies the date when the Activity actually started.
- planned_start_date : Event_or_date_select [0..1]
  The planned_start_date specifies the date when the Activity is or was supposed to be started.
- planned_end_date : Period_or_date_select [0..1]
  The planned_end_date specifies the date when the Activity is or was supposed to be finished.
- actual_end_date : Date_time [0..1]
  The actual_end_date specifies the date when the Activity actually finished.
- requestor : Date_and_person_organization [0..1]
  The requestor specifies the Person or Organization that requested the Activity and the date the request was submitted.
- supplying_organization : Organization [0..*]
  The supplying_organization specifies the set of Organization objects that carry out the work.
- concerned_organization : Organization [0..*]
  The concerned_organization specifies the set of Organization objects that are affected by the result of the Activity.
- resolved_request : Work_request [0..*]
  The resolved_request specifies the set of Work_request objects that are resolved by the Activity.

### 8.11.1.2 Class Activity_element

An Activity_element is an item of work that is part of an Activity.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1]
  The role specifies the function that is performed by the Activity_element in the context of the concerned Activity. Where applicable the following values shall be used:
  - 'control': The referenced element is an object that has immediate influence on the Activity performed;
  - 'input': The referenced element serves as initial data for the Activity;
  - 'output': The referenced element is a result of the Activity.

#### *Compositions*

- element_delivery : Element_delivery [0..*]
  The Element_delivery specifies the Element_delivery which this Activity_element is subject to.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Activity_element.

#### *Associations*

- element : Activity_element_select [1]The element specifies the piece of product data that is under work.

### 8.11.1.3 Class Activity_method

An Activity_method is a procedure that may be used to solve a request.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- consequence : String [0..1] The consequence specifies the expected positive or negative effects of the application of a particular Activity_method.

#### *Compositions*

- activity_method_assignment : Activity_method_assignment [0..*]
  The activity_method_assignment specifies the activity_method_assignment for which this activity_method is recommended or shall not be chosen.
- name : String_select [1] The name specifies the word or group of words by which the Activity_method is referred to.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Activity_method.

#### *Associations*

- description : String_select [1] The description specifies additional information that defines the Activity_method in terms of either the nature of the Activity_method or in terms of the specific procedure steps required to implement it.

### 8.11.1.4 Class Activity_method_assignment

An Activity_method_assignment is an object that associates an Activity_method with a Work_request. The associated Activity_method serves as a recommended or non-recommended method to resolve the tasks specified in the Work_request.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies whether the specified Activity_method may be used or not. Where applicable the following values shall be used:
  - 'non recommended method': The specified Activity_method shall not be used in order to accomplish the specified Work_request;
  - 'recommended method': The specified Activity_method may be used in order to accomplish the specified Work_request.

#### *Compositions*

- simple_property_value : Simple_property_value (ABS) [0..*]The simple_property_value specifies the assigned simple property values.

#### *Associations*

- associated_request : Work_request [1]
  The associated_request identifies the Work_request that the recommended or non-recommended method applies to.

### 8.11.1.5 Class Activity_relationship

An Activity_relationship is a relationship between two Activity objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'alternative': The application object defines a relationship where the related Activity may be used alternatively instead of the relating Activity;
  - 'decomposition': The application object defines a relationship where the related Activity is one of potentially more sub-activities into which the relating Activity is broken down;
  - 'derivation': The application object defines a relationship where the related Activity is derived from the relating Activity
  - 'exclusiveness': The application object defines a relationship where the relating and the related Activity shall not have any overlap in time of execution;
  - 'precedence': The application object defines a relationship where the related Activity has higher priority than the relating Activity;
  - 'sequence': The application object defines a relationship where the relating Activity shall be completed before the related Activity starts;
  - 'simultaneity': The application object defines a relationship that establishes that both the relating and related Activity are considered as occurring during the same time period or shall be performed together in order to ensure consistency and enhance efficiency.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Activity_relationship.

#### *Associations*

- related : Activity [1] The related specifies the second of the two Activity objects related by an Activity_relationship.

### 8.11.1.6 Class Change

A Change is a mechanism to collect the Model_change objects and the Property_change objects that describe the differences between the two objects referenced by the specified relationship object.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Change.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Change.

#### *Associations*

- none

### 8.11.1.7 Class Element_delivery

An Element_delivery is the specification of the expected delivery of an Activity_element.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- quantity : Value_with_unit (ABS) [1]
  The quantity specifies the number of objects referred by the Activity_element to be delivered.
- destination : Organization [1] The destination specifies the Organization the Activity_element is to be delivered to.

### 8.11.1.8 Class Project

A Project is an identified program of work.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Project.

#### *Compositions*

- Project_relationship : Project_relationship [0..*]
  The Project_relationship specifies the Project_relationship that relates the first of the two Project objects.
- description : String_select [0..1]The description specifies additional information about the Project.
- name : String_select [1] The name specifies the word or group of words by which the Project is referred to.
- document_assignment : Document_assignment [0..*]The document_assignment specifies the object that provides information for this Project.

#### *Associations*

- planned_end_date : Period_or_date_select [0..1]
  The planned_end_date specifies either the date when the Project is or was supposed to be finished or the planned duration of the Project.
- work_program : Activity [0..*] The work_program specifies the Activity objects that are carried out within the Project.
- planned_start_date : Event_or_date_select [0..1]The planned_start_date specifies the date when the Project is or was supposed to be started.
- actual_end_date : Date_time [0..1]The actual_end_date specifies the date when the Project was actually finished.
- actual_start_date : Date_time [0..1]The actual_start_date specifies the date when the Project was actually started.
- is_applied_to : Project_information_select [0..*]
  The is_applied_to specifies the set of objects that the work carried out by a Project applies to.

### 8.11.1.9 Class Project_relationship

A Project_relationship is a relationship between two Project objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'decomposition': The application object defines a relationship where the related Project is one of potentially more components into which the relating Project is broken down;
  - 'dependency': The related Project is dependent upon the relating Project;
  - 'sequence': The application object defines a relationship where the relating Project shall be completed before the related Project starts;
  - 'succession': The related Project is the successor of the relating Project.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Project_relationship.

#### *Associations*

- related : Project [1] The related specifies the second of the two Project objects related by a Project_relationship.

### 8.11.1.10 Class Work_order

A Work_order is the authorization for one or more Activity objects to be performed.

#### *Base Class*

- PLM_root_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Work_order.
- version_id : String [0..1] The version_id specifies the identification of a particular version of a Work_order.
- work_order_type : String [1] The work_order_type specifies the kind of the Work_order. Where applicable the following values shall be used:
  - 'design deviation permit': An authorization for a deviation from the approved design data;
  - 'design release': An authorization for the design of a product or of an item or to create a bill of material;
  - 'management resolution': An authorization by a committee, such as the board of directors, to design or change an item;
  - 'manufacturing release': An authorization for the manufacturing process of a product or of an item;
  - 'production deviation permit': An authorization for a deviation from the approved manufacturing process.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Work_order.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Work_order.

#### *Associations*

- is_controlling : Activity [1..*] The is_controlling specifies the Activity objects that are controlled by this particular Work_order.

### 8.11.1.11 Class Work_request

A Work_request is the solicitation for some work to be done.

#### *Base Class*

- PLM_root_object (ABS)

### *Attributes*

- id : String [1] — The id specifies the identifier of the Work_request.
- request_type : String [1] — The request_type specifies the intention of the Work_request. Where applicable the following values shall be used:
  - 'change of standard': A request to translate a change to a standard into action;
  - 'cost reduction': A request aimed at reducing the engineering and manufacturing costs of an item;
  - 'customer rejection': A request resulting from a rejection by a customer;
  - 'customer request': A request for an activity that is necessary to solve the request of a customer;
  - 'durability improvement': A request aimed at extending the life time of an item;
  - 'government regulation': A request resulting from legal requirements;
  - 'procurement alignment': A request to adjust the purchasing process of different items;
  - 'production alignment': A request to adjust the manufacturing process of different items;
  - 'production relief': A request aimed at achieving a simpler assembly and production process;
  - 'production requirement': A request for an activity that is necessary from a production point of view;
  - 'quality improvement': A request aimed at increasing the quality of an item;
  - 'security reason': A request for an activity that is necessary from a security point of view;
  - 'standardization': A request to unify variants of an item;
  - 'supplier request': A request for an activity necessary to solve the request of a supplier;
  - 'technical improvement': A request aimed at improving the technical aspects of an item;
  - 'tool improvement': A request aimed at increasing the useful life of a tool.
- status : String [1] — The status specifies the stage of the Work_request. Where applicable the following values shall be used:
  - 'in work': The request is being developed;
  - 'issued': The request has been completed and reviewed, and immediate action takes place;
  - 'proposed': The request has been completed and is awaiting review and authorization;
  - 'resolved': The request is resolved; the actions as defined by the request have been completed and no further work is required.
- version_id : String [0..1] — The version_id specifies the identification of a particular version of a Work_request.

### *Compositions*

- description : String_select [0..1]
  The description specifies additional information about the Work_request.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Work_request.

### *Associations*

- notified_person : Date_and_person_organization [1..*]
  The notified_person specifies the personnel that shall be informed about the Work_request and the date when the personnel or organization shall be informed.
- scope : Activity_element_select [0..*]
  The scope specifies the objects that are subject to the Work_request.

- requestor : Date_and_person_organization [1]

  The requestor specifies the person or organization who issued the Work_request and the date when this person or organization issued the Work_request.

### 8.11.2 Interfaces

#### 8.11.2.1 Interface Activity_element_select

This empty interface is realized by the following classes:

- Activity_method
- Specification_inclusion
- Specification_expression
- Specification_category
- Specification
- Product_structure_relationship
- Product_identification
- Product_class
- Physical_instance
- Manufacturing_configuration (ABS)
- Design_constraint
- Configuration
- Complex_product (ABS)
- Class_structure_relationship
- Class_specification_association
- Class_inclusion_association
- Class_condition_association
- Class_category_association
- Document_version
- Document_representation (ABS)
- Document_file (ABS)
- Geometric_model
- Document
- Item_version
- Item_definition_relationship (ABS)
- Item
- Design_discipline_item_definition

- Physical_assembly_relationship
- Item_instance_relationship (ABS)
- Item_instance (ABS)
- Item_definition_instance_relationship (ABS)
- Assembly_substitute_relationship
- Alternate_item_relationship
- General_classification
- Project
- Process_plan
- Process_operation_occurrence
- Process_operation_definition
- Property_value_association (ABS)
- Simple_property_association
- Property (ABS)
- Material
- Geometric_model

#### 8.11.2.2 Interface Change_relationship_select

***Compositions***

- change : Change [0..*]

***Implemented By***

- Process_operation_occurrence_relationship
- Process_plan_relationship
- Shape_element_relationship
- Design_constraint_relationship
- Replaced_definition_relationship
- Replaced_usage_relationship
- Complex_product_relationship
- Item_version_relationship

#### 8.11.2.3 Interface Project_information_select

This empty interface is realized by the following Classes:

- Product_identification
- Product_class

- Physical_instance
- Complex_product (ABS)
- Document_version
- Document
- Item_version
- Item_instance
- Item
- Activity

## 8.12 Package Process_planning

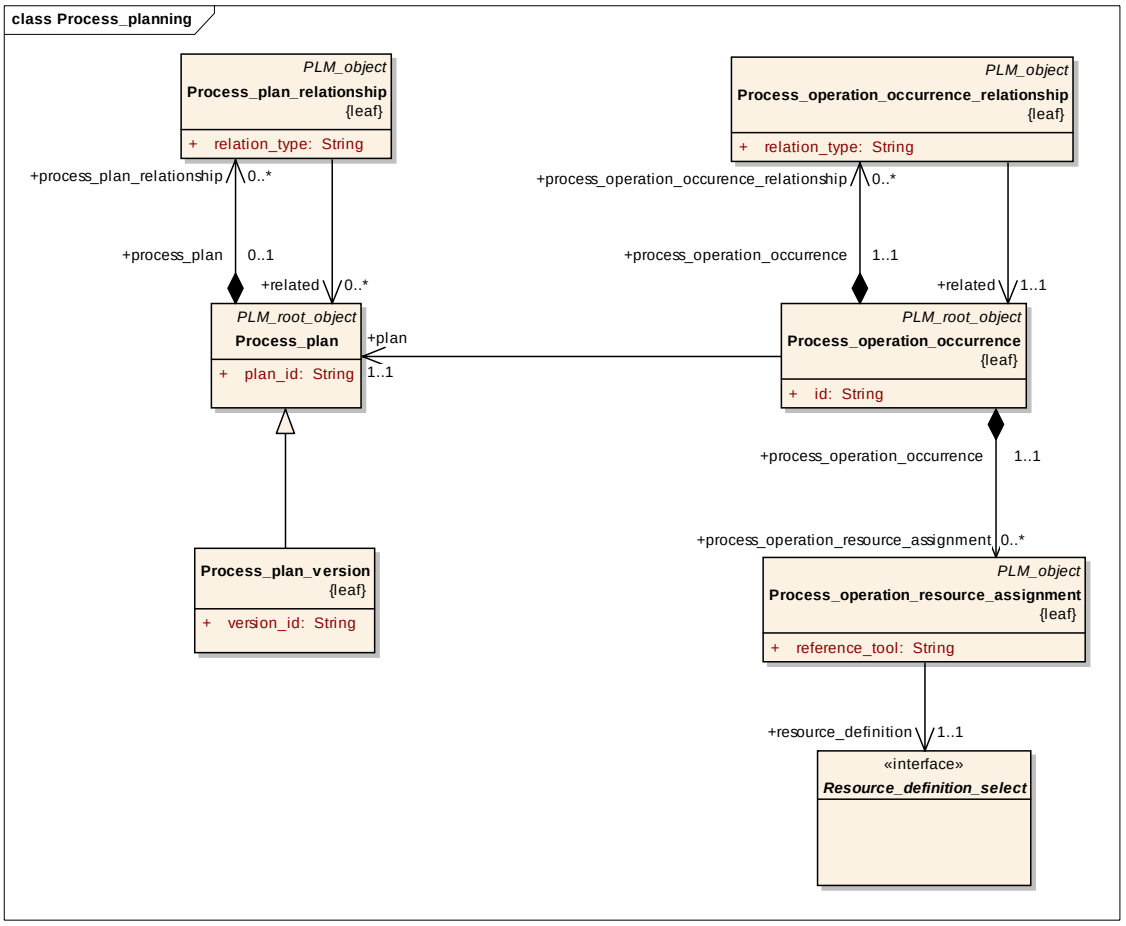


**Figure 8.43 - Process planning**

## 8.12.1 Classes

### 8.12.1.1 Class Mated_item_association

A Mated_item_association is a relationship between a Mating_definition and an Item_instance that is used within the Mating_definition.

A Mated_item_association is a type of Item_definition_instance_relationship.

#### *Base Class*

- Item_definition_instance_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- mated_item_relationship : mated_item_relationship[0..*]

#### *Associations*

- placement : Transformation_select[0..1]
  The placement specifies the Geometric_model_relationship_with_transformation or the Template_instance that conveys the Transformation information. This Transformation is used to locate and to orient the constituent in the coordinate space of the Mating_definition. In the case of a Template_instance, the scale factor shall be omitted or set to 1.0.

### 8.12.1.2 Class Mated_item_relationship

A Mated_item_relationship is a relationship between two Mated_item_association objects.

This relationship specifies additional information about the mating of two particular items that go into a Mating_definition. The two Mated_item_association objects that are referenced by the Mated_item_relationship shall refer to the same Mating_definition.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- mated_shape : Shape_element_relationship[0..1]
  The mated_shape specifies the Shape_element_relationship that relates the two Shape_element objects that form the area of mating contact prior to mating.

- mating_material : Quantified_instance[0..1]
  The mating_material specifies the set of Quantified_instance objects used as material for the mating.
- related : Mated_item_association[1]
  The related specifies the second of the two Mated_item_association objects related by the Mated_item_relationship.

#### 8.12.1.3 Class Mating_definition

A Mating_definition is a view of an Item_version, defining the physical connection of two or more Item_instance objects. It includes technical information about the kind of connection. This information is independent from the hierarchical assembly structure.

A Mating_definition is a type of Design_discipline_item_definition.

##### *Base Class*

- Design_discipline_item_definition

##### *Attributes*

- mating_type: string[1] The mating_type specifies the kind of mating, i.e., how the items shall be mated together.

##### *Compositions*

- mated_items: mated_item_association [2..*]

##### *Associations*

- none

#### 8.12.1.4 Class Process_operation_definition

A Process_operation_definition is the specification of an activity that may be included in a Process_plan. A Process_operation_definition characterizes a manufacturing or control operation.

##### *Base Class*

- PLM_root_object (ABS)

##### *Attributes*

- id : String [1] The id specifies the identifier of the Process_operation_definition that shall be unique within the scope of the associated Process_plan_version.
- process_type : String [1] The process_type specifies the type of the Process_operation_definition.
- version_id : String [0..1] The version_id specifies the identification of a particular version of a Process_operation_definition.

##### *Compositions*

- process_operation_definition_relationship : Process_operation_definition_relationship [0..*]
  The process_operation_definition_relationship specifies the

process_operation_definition_relationship that relates the first of the two Process_operation_definition objects.

- description : String_select [0..1]The description specifies additional information about the Process_operation_definition.
- name : String_select [0..1] The name specifies the word or group of words by which the Process_operation_definition is referred to.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- none

### 8.12.1.5 Class Process_operation_definition_relationship

A Process_operation_definition_relationship is a relationship between two Process_operation_definition objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'alternative': The application object defines a relationship where the related Process_operation_definition may be used alternatively instead of the relating Process_operation_definition;
  - 'substitution': The application object defines a relationship where the related Process_operation_definition replaces the relating Process_operation_definition;
  - 'version association': The application object defines a relationship where the related Process_operation_definition is a version of the relating Process_operation_definition. In this case, only the related Process_operation_definition shall specify a version_id.;
  - 'version sequence': The application object defines a relationship where the relating Process_operation_definition is the preceding version and the related Process_operation_definition is the following version. In this case, both Process_operation_definition objects shall specify a version_id.

#### *Compositions*

- none

#### *Associations*

- related : Process_operation_definition [1]
  The related specifies the second of the two objects related by the Process_operation_definition_relationship.

### 8.12.1.6 Class Process_operation_input_or_output

A Process_operation_input_or_output is the input or expected result of a Process_operation_definition.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- role : String [1] The role specifies whether the identified element plays the role of an input or an output for the operation.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Process_operation_input_or_output.

#### *Associations*

- concerned_shape : Shape_element [0..*]The concerned_shape specifies the set of Shape_element objects that are affected by the Process_operation_occurrence.
- placement : Transformation (ABS) [0..1]
  The placement specifies the geometrical Transformation between the local coordinate system of the element acting as Process_operation_input_or_output, and the reference coordinate system. The reference coordinate system is either the coordinate system of the reference tool, if present, for the concerned Process_operation_occurrence or, if no reference tool is present, the coordinate system of the Process_operation_occurrence itself.
- element : Process_operation_input_or_output_select [1]
  The element specifies the element that plays the role of the input or the output for the operation.

### 8.12.1.7 Class Process_operation_occurrence

A Process_operation_occurrence is the usage of a Process_operation_definition in a Process_plan. This association states that the Process_operation_definition is part of the Process_plan.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- id : String [1] The id specifies the identifier of the Process_operation_occurrence.

#### *Compositions*

- process_operation_resource_assignment : Process_operation_resource_assignment [0..*]
  The process_operation_resource_assignment specifies the process_operation_resource_assignment that is associated with this Process_operation_occurrence.
- process_operation_occurence_relationship : Process_operation_occurrence_relationship [0..*]
  The process_operation_occurence_relationship specifies the process_operation_occurence_relationship that relates the first of the two Process_operation_occurrence objects.

- process_operation_input_or_output : Process_operation_input_or_output [0..*]
  The process_operation_input_or_output specifies the process_operation_input_or_output that s associated with this Process_operation_occurrence.
- configuration : Configuration [0..*]
  The configuration specifies the configuration that controls this Process_operation_occurrence for its valid usage.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Process_operation_occurrence.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- operation_definition : Process_operation_definition [1]The operation_definition specifies the Process_operation_definition that defines the Process_operation_occurrence in a Process_plan.
- is_defined_in : Cartesian_coordinate_space (ABS) [0..1]
  The is_defined_in specifies the Cartesian_coordinate_space of the Process_operation_occurrence for the case where none of the tools associated by Process_operation_input_or_output plays the role of a reference tool defining the reference coordinate space.
- plan : Process_plan [1] The plan specifies the Process_plan to which the Process_operation_occurrence is assigning a Process_operation_definition.

### 8.12.1.8 Class Process_operation_occurrence_relationship

A Process_operation_occurrence_relationship is a relationship between two Process_operation_occurrence objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'decomposition': The application object defines a relationship where the related Process_operation_occurrence is one of the components of the relating Process_operation_occurrence;
  - 'exclusiveness': The application object defines a relationship where the relating and the related Process_operation_occurrence shall not have any overlap in time of execution;
  - 'sequence': The application object defines a relationship where the relating Process_operation_occurrence shall be completed before the related Process_operation_occurrence starts;
  - 'simultaneity': The application object defines a relationship where the relating and the related Process_operation_occurrence are considered as occurring during the same time period;
  - 'substitution': The application object defines a relationship where the related Process_operation_occurrence replaces of the relating Process_operation_occurrence.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Process_operation_occurrence_relationship.
- change : Change [0..*] The change specifies the change for which this object references a modified object and the corresponding original object.

#### *Associations*

- cycle_time : Duration [0..1] The cycle_time specifies the interval of time within which both Process_operation_occurrence objects have to take place in order to be declared as simultaneous.
- waiting_time : Property_value (ABS) [0..1]
  The waiting_time specifies the time which shall elapse, at least, between the completion of the relating Process_operation_occurrence and the start of the related Process_operation_occurrence. The referenced shall have a definition that is a Duration_property.
- related : Process_operation_occurrence [1]
  The related specifies the second of the two Process_operation_occurrence objects related by a Process_operation_occurrence_relationship.

### 8.12.1.9 Class Process_operation_resource_assignment

A Process_operation_resource_assignment is a mechanism to associate a resource with a Process_operation_occurrence.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- reference_tool : Boolean [1] The reference_tool specifies whether or not the resource identified by the Process_operation_resource_assignment plays the role of the reference tool for the occurrence of an operation.

#### *Compositions*

- reason : String_select [0..1] The reason specifies the rationale behind the use of the resource for a particular Process_operation_occurrence.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

#### *Associations*

- placement : Transformation (ABS) [0..1]
  The placement specifies the geometrical Transformation between the local coordinate system of the Process_operation_resource_assignment and the reference coordinate system.
- resource_definition : Resource_definition_select [1]
  The resource_definition specifies the tool that is used to perform the operation.

#### 8.12.1.10 Class Process_plan

A Process_plan is the manufacturing planning information, necessary to realize or produce a particular version of an Item.

##### *Base Class*

- PLM_root_object (ABS)

##### *Attributes*

- plan_id : String [1] The plan_id specifies the identifier of the Process_plan that shall be unique within the scope of an organization.

##### *Compositions*

- process_plan_relationship : Process_plan_relationship [0..*]
  The process_plan_relationship specifies the process_plan_relationship that relates the first of the two Process_plan objects.
- description : String_select [0..1]The description specifies additional information about the Process_plan.
- name : String_select [0..1] The name specifies the word or group of words by which the Process_plan is referred to.
- configuration : Configuration [0..*]
  The configuration specifies the configuration that controls this Process_plan for its valid usage.
- document_assignment : Document_assignment [0..*]
  The document_assignment specifies the object that provides information for this Process_plan.
- simple_property_value : Simple_property_value (ABS) [0..*]
  The simple_property_value specifies the assigned simple property values.

##### *Associations*

- produced_output : Item_version [0..*]
  The produced_output specifies the set of Item_version objects that are produced by the operations of the Process_plan.

#### 8.12.1.11 Class Process_plan_relationship

A Process_plan_relationship is the relationship between two Process_plan objects.

##### *Base Class*

- PLM_object (ABS)

##### *Attributes*

- relation_type : String [1] The relation_type specifies the meaning of the relationship. Where applicable the following values shall be used:
  - 'alternative': The application object defines a relationship where the related Process_plan may be used alternatively to the relating Process_plan;
  - 'version association': The application object defines a relationship where the related Process_plan is a version of the relating Process_plan. In this case, the related

Process_plan shall be a Process_plan_version;
- 'version sequence': The application object defines a relationship where the relating Process_plan is the preceding version and the related Process_plan is the following version. In this case, both Process_plan objects shall be of type Process_plan_version.

#### *Compositions*

- description : String_select [0..1]The description specifies additional information about the Process_plan_relationship.
- change : Change [0..*] The change specifies the change for which this object references a modified object and the cor-responding original object.

#### *Associations*

- related : Process_plan [1] The related specifies the second of the two Process_plan objects related by a Process_plan_relationship.

### 8.12.1.12 Class Process_plan_version

A Process_plan_version is a particular version of a Process_plan.

#### *Base Class*

- Process_plan

#### *Attributes*

- version_id : String [1] The version_id specifies the identification of a particular version of a Process_plan.

#### *Compositions*

- none

#### *Associations*

- none

### 8.12.1.13 Class Process_property_association

A Process_property_association is a mechanism to assign a property value to process related objects.

#### *Base Class*

- Property_value_association (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- described_element : Process_property_select [1]
  The described_element specifies the object that is described by the property value.

### 8.12.1.14 Class Process_state

A Process_state is a view of an in-process-item definition of a particular version of an Item. It characterizes a state of the Item_version that occurs before the state identified by the 'related_item_definition'. The identifier of a Process_state shall be unique within the context of the Item_version and of the Process_plan_version.

#### *Base Class*

- Design_discipline_item_definition

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- related_item_definition : Design_discipline_item_definition [1]
  The related_item_definition specifies the Design_discipline_item_definition that defines the final item that the in-process-item is a preliminary stage of.

### 8.12.1.15 Class Process_state_relationship

A Process_state_relationship is a relationship between two Design_discipline_item_definition objects where the relating Design_discipline_item_definition is view of an in-process-item definition of a particular version of an Item. It characterizes a state of the Item_version that occurs before the state identified by the related Design_discipline_item_definition.

**Note –** Process_state_relationship may be used to characterize intermediate states of the Item_version, if this Item_version is one of the produced outputs of a Process_plan.

A Process_state_relationship is a type of Item_definition_relationship.

#### *Base Class*

- Item_definition_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- description: String_select[0..1] The description specifies additional information about the Process_state_relationship.

#### *Associations*

- none

#### 8.12.1.16 Class Same_time_machining_relationship

A Same_time_machining_relationship is a relationship between two Item_instance objects that specifies that those two objects shall be manufactured together within the same Process_operation_occurrence.

A Same_time_machining_relationship is a type of Item_instance_relationship.

#### *Base Class*

- Item_instance_relationship (ABS)

#### *Attributes*

- none

#### *Compositions*

- description : String_select[0..1]
  The description specifies additional information about the Same_time_machining_relationship.

#### *Associations*

- placement : Transformation_select[0..1]
  The placement specifies the geometrical Transformation between the local coordinate system of the related Item_instance with respect to the local coordinate system of the relating Item_instance. NOTE The transformation is needed to position the related Item_instance with respect to the relating Item_instance for the same time machining process. In the case where the placement is realized by a Template_instance the scale factor of that Template_instance shall be omitted or set to 1.

### 8.12.2 Interfaces

#### 8.12.2.1 Interface Process_operation_input_or_output_select

This empty interface is realized by the following classes:

- Design_discipline_item_definition
- Item_instance (ABS)
- Assembly_component_relationship
- Mated_item_relationship

#### 8.12.2.2 Interface Process_property_select

This empty interface is realized by the following classes:

- Activity_method_assignment
- Activity

- Process_plan
- Process_operation_resource_assignment
- Process_operation_occurrence
- Process_operation_definition

#### 8.12.2.3 Interface Resource_definition_select

This empty interface is realized by the following classes:

- Product_component
- Physical_instance
- Descriptive_specification
- Design_discipline_item_definition
- Item_instance (ABS)

## 8.13 Package Multi_language_support

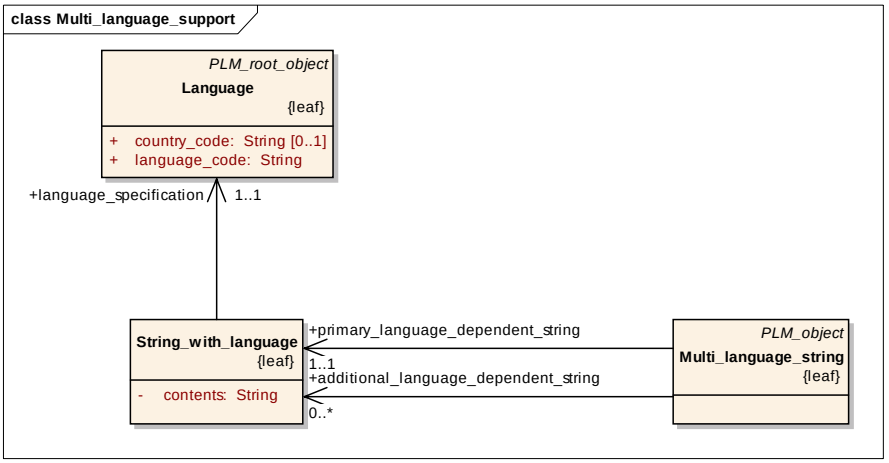


**Figure 8.44 - Multi language support**

### 8.13.1 Classes

#### 8.13.1.1 Class Language

A Language is a specification of the language in which an information is given.

***Base Class***

- PLM_root_object (ABS)

#### *Attributes*

- language_code : String [1] The language_code specifies the language of the text information in the Alpha-3 bibliographic code specified in ISO 639-2.
- country_code : String [0..1] The country_code specifies the country, as addition to the language, according to the alpha-2 code specified in ISO 3166-1.

#### *Compositions*

- none

#### *Associations*

- none

### 8.13.1.2 Class Multi_language_string

A Multi_language_string represents text information, expressed in one or more languages, that is associated with objects.

#### *Base Class*

- PLM_object (ABS)

#### *Attributes*

- none

#### *Compositions*

- none

#### *Associations*

- primary_language_dependent_string : String_with_language [1]
  The primary_language_dependent_string specifies the String_with_language that represents the text information in the original language.
- additional_language_dependent_string : String_with_language [0..*]
  The additional_language_dependent_string specifies the String_with_language objects that represent the text information in a particular language.

### 8.13.1.3 Class String_with_language

A String_with_language represents text information in a specific language together with an identification of the language used.

#### *Base Class*

- none

#### *Attributes*

- contents : String [1] The contents is textual information stored in the language identified by the language attribute.

***Compositions***

- none

***Association***

- language_specification : Language [1]
  The language_specification specifies the Language in which the contents is given.

## 8.13.2 Interfaces

### 8.13.2.1 Interface String_select

This empty interface is realized by the following class:

- Multi_language_string

## 8.13.3 Datatypes

### 8.13.3.1 Datatype Default_language_string

# 9 Computational Viewpoint (normative)

## 9.1 Overview

The computational viewpoint captures the functional aspects of the model described in Section 8 . There are many different use-cases for the platform independent data model. The main usage of STEP ISO 10303-214:214 [8] is the exchange of engineering data, but nowadays some companies think about using STEP as a company wide data model for all information exchange process.

To support a wide range of use cases the data model must be enriched by functional elements. Those elements should support an effective and easy to use interface for handling the data model.

The Computational Viewpoint provides the necessary life cycle functionality to create, read, update and possibly to delete instances of the data model defined in the Informational Viewpoint. Especially, it defines a mechanism to query and traverse instances of the Informational Viewpoint. Therefore, the Computational Viewpoint is dependent on the Informational Viewpoint.

## 9.2 Base Model of the Computational Viewpoint

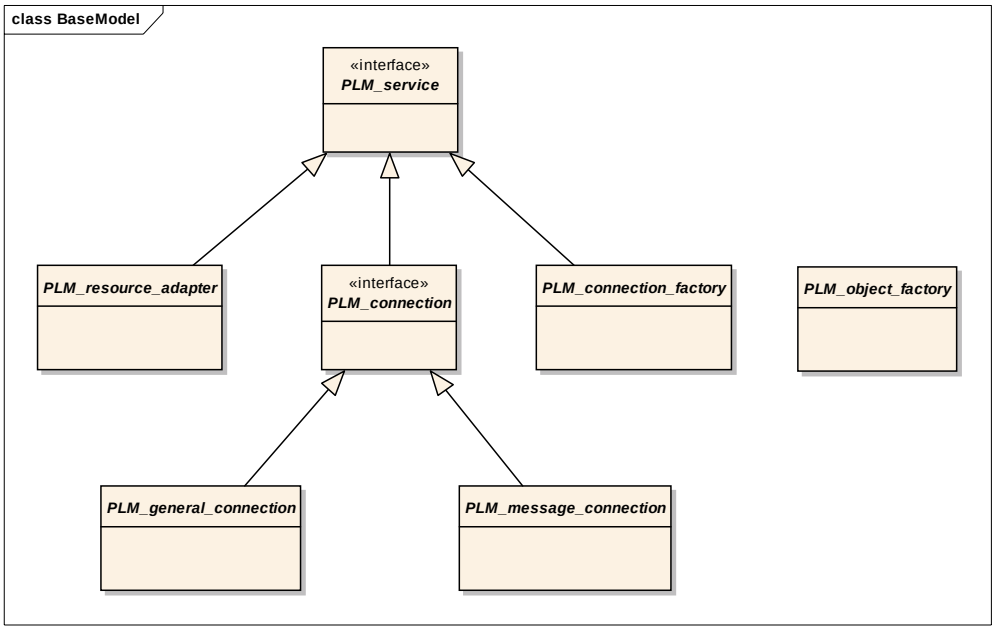


**Figure 9.1 - Base Model of the Computational Viewpoint**

The Computational Viewpoint has a similar functional model as a connector defined in the J2EE Connector Architecture specification. It uses five specific object types: PLM_resource_adapter, PLM_object_factory, PLM_connection_factory, PLM_general_connection and PLM_message_connection, the base interfaces PLM_service and PLM_connection and the data types URL, UID, PLM_query, PLM_core_container, PLM_message, PLM_context, PLM_processing_instruction, PLM_properties_descriptor and PLM_property.

The types PLM_core_container and UID are defined in the Informational Viewpoint. The type URL is used to model URL's. All operations of all interfaces can throw PLM_exception objects.

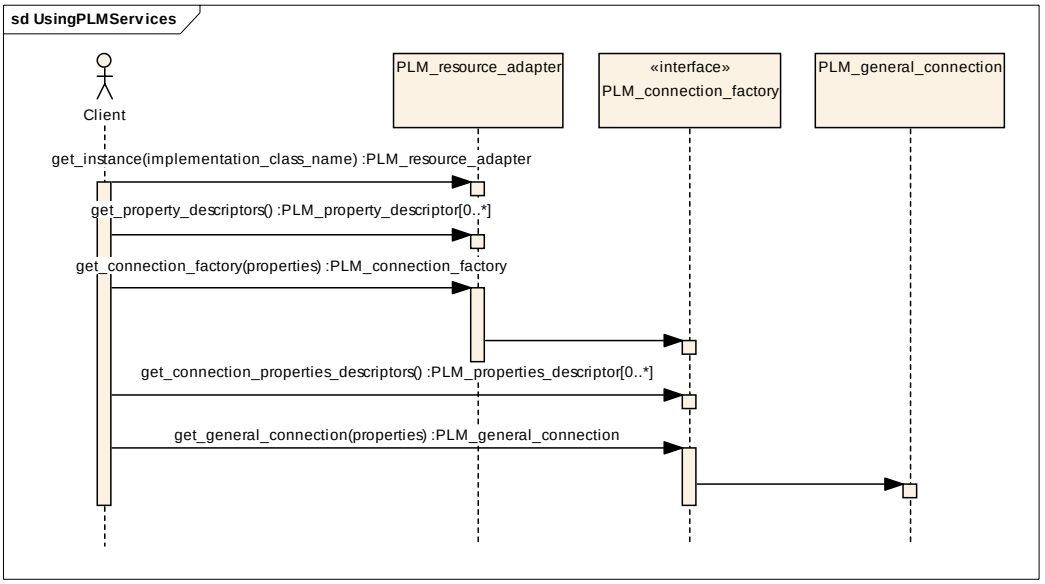


**Figure 9.2 - Using the Computational Viewpoint**

In Figure 9.2 a typical usage scenario is described which comprises the steps

- referring to a resource adapter service,
- obtaining invocation property informations,
- retrieving a suitable connection_factory,
- inspecting the general_connection properties and
- invoking a get on the factory instance.

### 9.2.1 PLM_service Interface

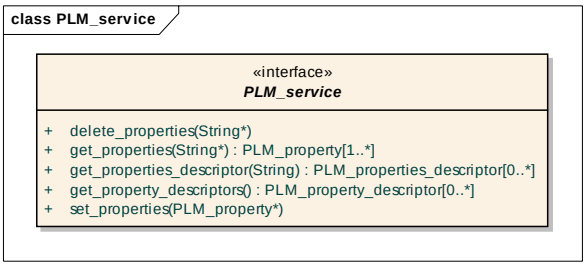


**Figure 9.3 - The PLM_service interface**

The PLM_service defines basic operations to set, get, delete, and get descriptions of properties of a service. It is the base definition for all services of the Computational Viewpoint.

#### *Set_properties Operation*

set_properties(in properties: PLM_property[]): void

The operation set_properties() sets new values for service properties.

#### *Get_properties Operation*

get_properties(in names: String[]): PLM_property []

The operation get_properties() returns the properties whch names are given in the input parameter names.

#### *Delete_properties Operation*

delete_properties(in names: String[]): void

The operation set_properties() sets new values for service properties.

#### *Get_property_descriptors Operation*

get_property_descriptors(): PLM_property_descriptors[]

The operation get_property_descriptors() returns descriptors for the properties provided by a service.

#### *Get_properties_descriptors Operation*

get_properties_descriptors(operation_name: String): PLM_properties_descriptor[]

The operation get_properties_descriptors() returns descriptors for all allowed property combination variants for a service operation which provides a set of PLM_property elements as parameter.

### 9.2.2 PLM_resource_adapter Class

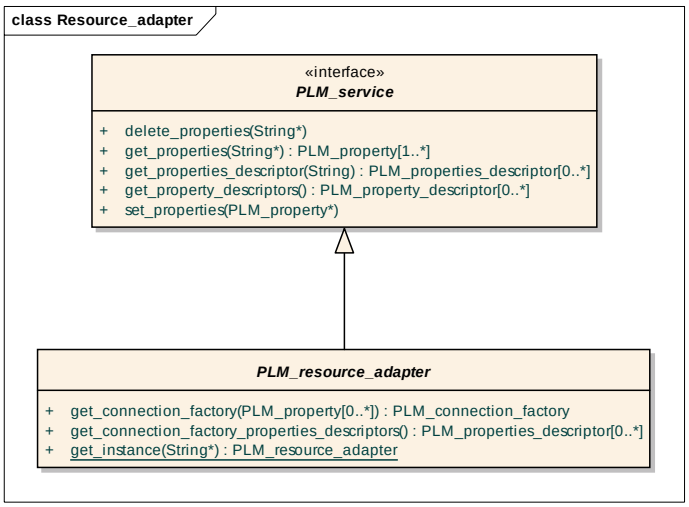


**Figure 9.4 - The PLM_resource_adapter Class**

A PLM connector vendor must provide an implementation of the abstract PLM_resource_adapter class. A client may obtain an instance of a specific PLM resource adapter class by the static member function get_instance() with the class name of the specific PLM resource adapter as parameter.

By the operation get_connection_factory() the client can obtain a PLM_connection_factory object. The value of the parameter name is the name of the PLM connection factory. The list of all supported values for this parameter can be obtained by the operation get_connection_factory_properties_descriptors(). In the parameter properties the client can pass specific parameters. The values and semantics of the properties parameter will be defined in the Platform Specific Models. Examples for property names are "java.naming.provider.url" and "java.naming.factory.initial" if the PLM connector implementation uses a JNDI name service.

### 9.2.3 PLM_object_factory Interface

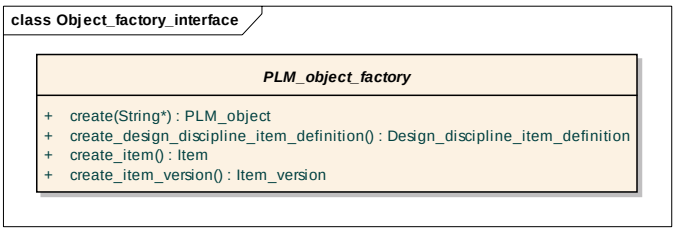


**Figure 9.5 - The PLM_object_factory Interface (fragmentary)**

The PLM_object_factory provides one specific create operation for each non abstract type which extends directly or indirectly PLM_object. Additionally a generic create operation is provided. Allowed parameter values for the generic create operation are the names of those types for which a specific create operation in the PLM_object_factory exists. The result PLM_objects from the create operations are local objects. The operation write() from the interface PLM_general_connection has to be used to transfer a local object to a PLM system (create a new object in the PLM system).

### 9.2.4 PLM_connection_factory Interface

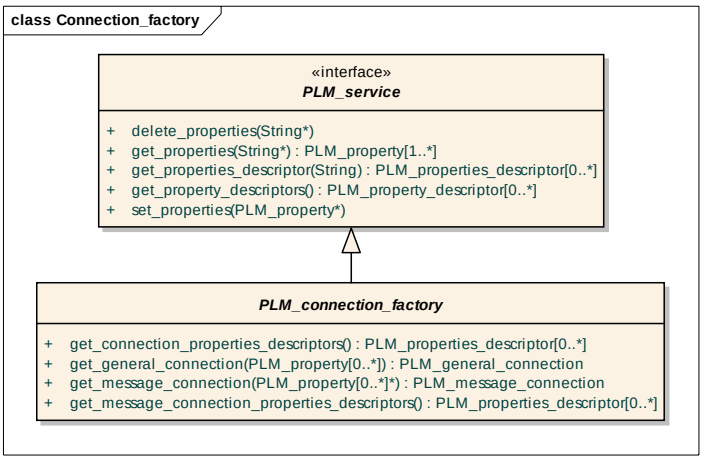


**Figure 9.6 - The PLM_connection_factory**

The interface PLM_connection_factory  specializes the interface PLM_service (see Figure 9.6 ) and provides the operation get_general_connection() which returns a PLM_general_connection instance and the operation get_message_connection() wich returns a PLM_message_connection instance. By the parameter properties the client may pass specific information to the PLM_connection_factory. The actual properties are implementation specific and its descriptors can be obtained by the operation get_properties_descriptor() and get_property_descriptors().

### 9.2.5 PLM_connection Interface

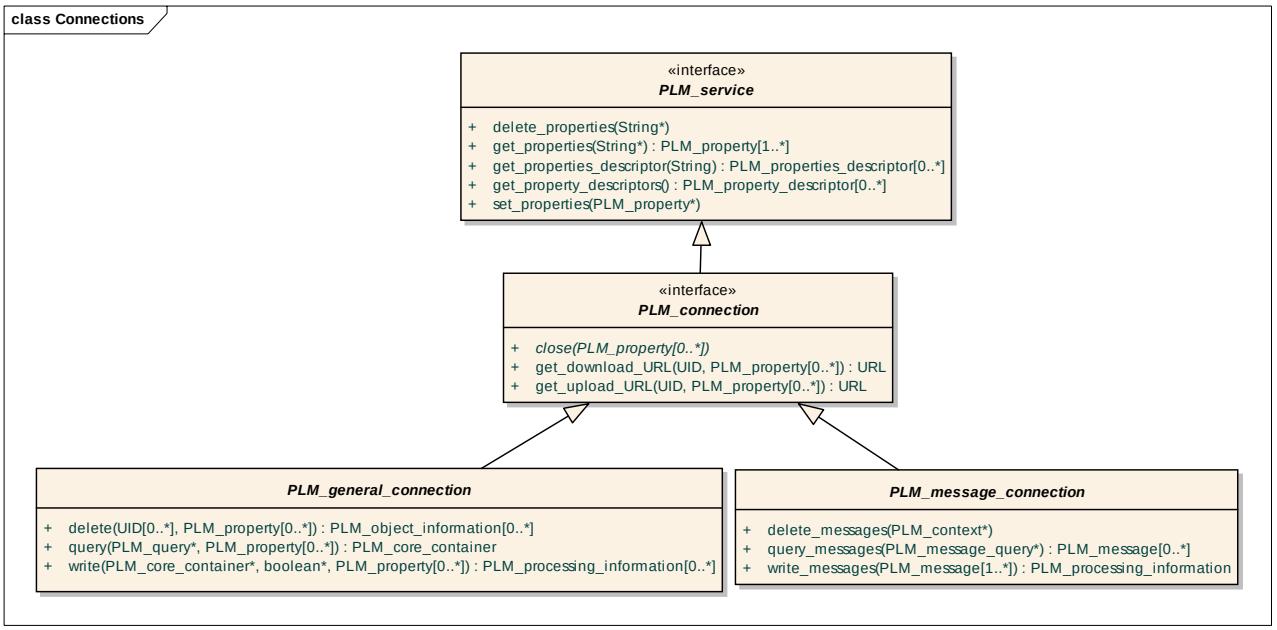


**Figure 9.7 - The PLM_connection, PLM_general_connection and PLM_message_connection**

The PLM_connection is derived from PLM_Service (see Figure 9.7 ) and is the base interface for PLM_general_connection and PLM_message_connection and defines their common operations.

#### *Get_download_URL Operation*

get_download_URL(in file_uid: UID, properties: PLM_property[0..*]): URL

The get_download_URL() operation is assigned an uid-attribute of a Digital_file-object as the first parameter. As a return value, it delivers an URL to retrieve the content of a Digital_file from the PLM system.

The get_download_URL() operation accepts a set of PLM_property elements as additionally parameter. The list of all supported values for this parameter can be obtained by the operation get_operation_properties_descriptors().The values and semantics of the properties parameter are implementation specific. One possible use of the properties parameter is to control the protocol of the returned URL, (e.g., "http", "ftp", "https").

#### *Get_upload_URL Operation*

get_upload_URL(in file_uid: UID, properties: PLM_property[0..*]): URL

The get_upload_URL() operation expects an uid-attribute of a Digital_file-object as the first parameter. It returns an URL which is used to upload a new content of the Digital_file to the PLM system.

The get_upload_URL() operation accepts a set of PLM_property elements as additionally parameter. The list of all supported values for this parameter can be obtained by the operation get_operation_properties_descriptors().The values and semantics of the properties parameter are implementation specific. One possible use of the properties parameter is to control the protocol of the returned URL, (e.g., "http," "ftp," "https").

#### *Close Operation*

close(properties: PLM_property[0..*]): void

The close() operation shuts down a connection to a PLM system. After a successful call of the close operation, all subsequent calls to this connection may raise an exception.

The close() operation accepts a set of PLM_property elements as parameter. The list of all supported values for this parameter can be obtained by the operation get_operation_properties_descriptors().The values and semantics of the properties parameter are implementation specific. One possible use of the properties parameter is to control session cleanup efforts on the PLM system.

### 9.2.6 PLM_general_connection

The PLM_general_connection is derived from the PLM_connection_interface (see Figure 9.7 ) and is the central service of this specification for a communication based an the request/respond approach. Its purpose is to grant access to the PLM system. To pass PLM data, it uses instances of the class PLM_container. To define the semantics of the operations, it is assumed, that all PLM data in the PLM system is instantiated as a single instance of PLM_container and the implementation of the operations works on that instance.

#### *Query Operation*

query(in query: PLM_query, properties : PLM_property[0..*]): PLM_core_container

The operation query() expects a PLM_query instance as its input parameter query. By applying this query to the data in a PLM system, a set of selected nodes is generated. As result of the query, a PLM_core_container instance is returned containing all selected nodes of the query and all nodes required to fulfill the minimum multiplicity constraints of the relationships of the selected nodes. The query operation accepts a set of PLM_property objects as additional parameter. The allowed values and the semantic of this parameter are implementation specific and can be obtained by the operation get_operation_properties_descriptors().

#### *Write Operation*

write(data: PLM_core_container, fill_result_list: Boolean, properties: PLM_property[0..*]): PLM_processing_information[0..*]

The operation write() expects a PLM_core_container instance as an input parameter. The PLM system uses the uid-Attributes of the single nodes in the PLM_core_container instance to identify which nodes already exist in the PLM System and which nodes have to be created. The operation has a return value of PLM_processing_information objects. In this return value information on manipulated objects is given. If the client ignores this information, the parameter fill_result_list shall be set to FALSE. By creating a new node, it is for a PLM system in general not feasible to use the attributes of the parameter data set. The operation adds one Object_changed_information for each changed object. If the uid-Attribute, the id-Attribute (e.g., id,

name, document_id, file_id) or any other attribute has changed the new_object_uid, the new_object_id or the remainder_unchanged attributes of the Object_changed_information are set accordingly. The result list is also used to inform the client, if not all objects of the data parameter were inserted in the PLM System. This information is added to the result list as Object_not_inserted_information instances. It is allowed to an implementation of the operation write() to add extra PLM_objects such as creator or creation time objects to the PLM system. If a write operation adds additional PLM_objects to the PLM system, this information has to be added to the result list as Additional_objects_written_information instances.

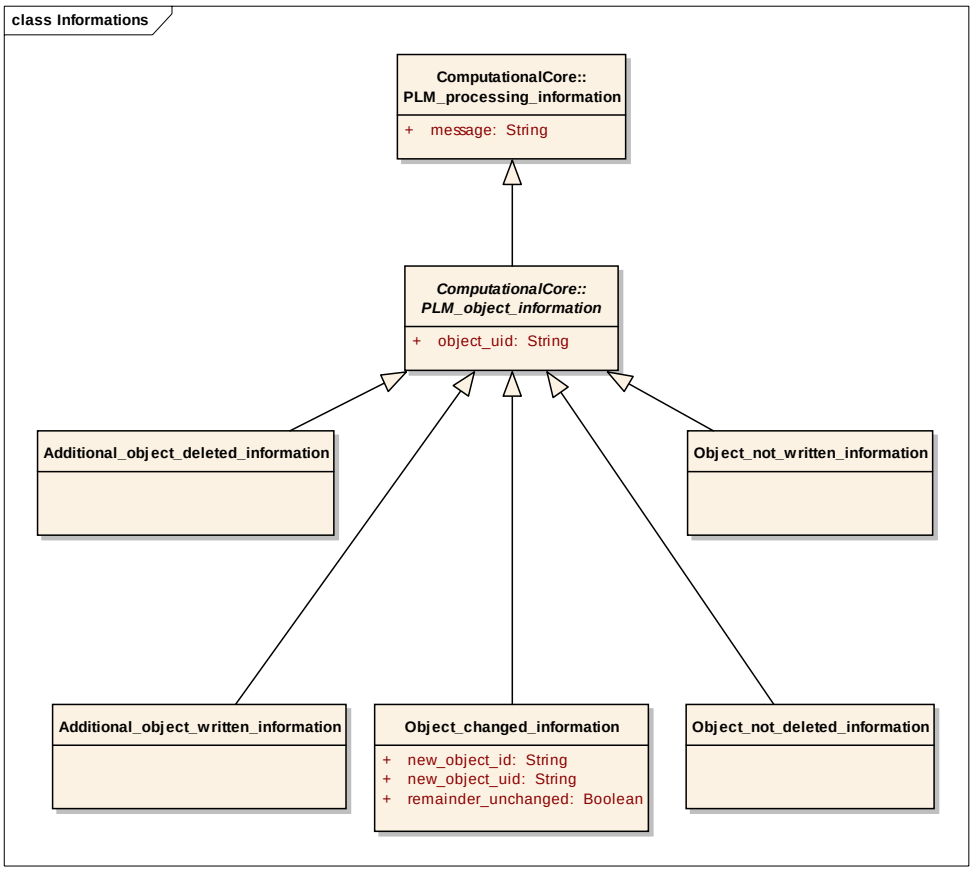


**Figure 9.8 - PLM_processing_information types of the result list of the write, import_data and delete operations**

All elements of the data set are transferred to the PLM system. Should one element already exist, all attribute values of the existing entity in the PLM system are replaced by the attributes values of the entity in the parameter. The relationships of an existing entity are not replaced by the relationships of the corresponding entity in the parameter. Instead, the relationships of the entity of the parameter not already existing are created.

The write operation accepts a set of PLM_property objects as additional parameter. The allowed values and the semantic of this parameter are implementation specific and can be obtained by the operation get_operation_properties_descriptors(). For example if the PLM system shall transform, filter or extent the input data prior writing to its data base this behaviour could be controlled by the properties parameter.

### *Delete Operation*

delete(in uids: UID[0..*]): PLM_processing_information[0..*]

The operation delete() expects a list of UID elements as input parameter. All objects with the given uids are deleted from the PLM system by this delete operation. If objects could not be deleted for some reasons, for each of those objects an Object_not_deleted_message instance that describes the reason has to returned in the result list. Additionally, all nodes are deleted, which no longer fulfill the minimum multiplicity constraints of their type. For each additionally deleted object an Additional_object_deleted_message instance has to be added to the result list.

The operation delete accepts a set of PLM_property objects as additional parameter. The allowed values and the semantic of this parameter are implementation specific and can be obtained by the operation get_operation_properties_descriptors(). For instance if the PLM system provides a recursive entity deletion capability, this functionality can be made available by a specific property parameter to the clients.

### 9.2.7 PLM_message_connection

The PLM_message_connection is derived from the PLM_connection_interface (see Figure 9.7 ) and is the central service of this specification for a communication based an the message exchange approach. It provides operations to query messages from a service, to write messages to a service and to delete messages from a service.

#### *Query_messages Operation*

Query_messages(in query: PLM_message_query): PLM_message[]

The query_messages() operation allows a client to receive messages from a services. It expects a PLM_message_query as the parameter. For the message exchange based approach of client server communication workflows must be specified which define when a client can or must send a new message to the service an when it can expect to receive a message from the service.

#### *Write_messages Operation*

Write_messages(in messages: PLM_message[]): PLM_processing_information[]

The write_messages() operation allows a client to send a message to a service. It expects one or more PLM_message-object(s) as the parameter. The processing of a message in the service depends on the type of the message. For the message exchange based approach of client server communication workflows must be specified which define when a client can or must send a new message to the service an when it can expect to receive a message from the service.

#### *Delete_messages Operation*

Delete_messages(in contexts: PLM_context[]): PLM_processing_information[]

The delete_messages() operation allows a client to delete explicitly messages in message queue on a service. It a expects one or more PLM_context-object(s) as the parameter to identify the messages to be deleted. If a client can or must explicitly delete a message from the message queue depends on the message type and on the workflow in which the message is used.

### 9.2.8 PLM_property_descriptor and PLM_properties_descriptor

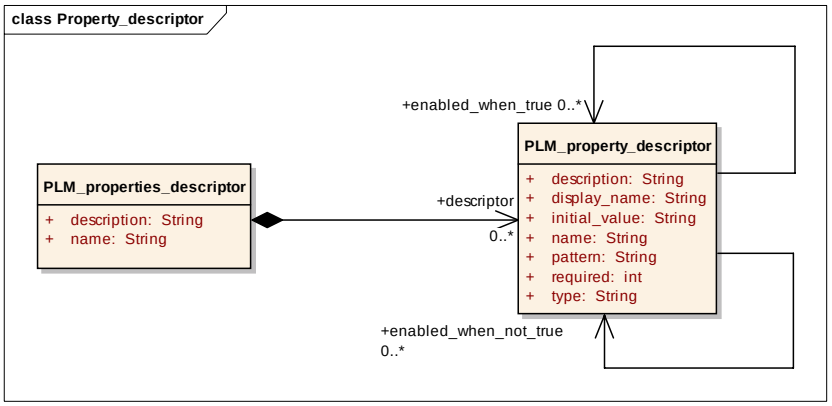


**Figure 9.9 - PLM_properties_descriptor and PLM_property_descriptor**

Some of the operations defined in the computational model use parameters of type PLM_property. The supported values of those parameters are implementation specific. Each operation with a parameter of type PLM_property has a corresponding operation which a client can use to obtain descriptions of the actual supported variants of values of the properties parameter. One supported variant of values is described by an instance of type PLM_properties_descriptor. A PLM_properties_descriptor has an attribute name that contains the name of the variant, an attribute description which contains a description of the variant and a list of PLM_property_descriptor. Each element of the PLM_properties_descriptor list describes one PLM_property of the variant.

A PLM_property_descriptor describes one PLM_property instance. The attribute name of the PLM_property_descriptor defines the value of attribute name of the PLM_property instance. The attribute type describes the type of the PLM_property instance. The attribute pattern defines a pattern that must match valid values of attribute value of the PLM_property. The attribute description of the PLM_property_descriptor contains a description of the described PLM_property instance. The attribute required defines wether the described PLM_property instance must be present or if it is optional. The references enabled_when_true and enabled_when_not_true can select other PLM_property_descriptor instances. The selected instance must have the type Boolean and must be contained by the same PLM_properties_descriptor instance. If a PLM_property_descriptor has enabled_when_true- or enabled_when_not_true references its attribute required must not have a value of TRUE. A described PLM_property value can only be used in a properties parameter list for an operation

- if all PLM_property values described by the PLM_property_descriptors referenced by enabled_when_true are also in the properties parameter list and have the value TRUE
- and no PLM_property_value described by a PLM_property_descriptor referenced by enabled_when_not_true is in the properties parameter list and has the value TRUE.

The value of the attribute display_name can be used as display name of the described PLM_property in user interfaces.

### 9.2.9 PLM_message_query

The PLM_message_query is the abstract base class for all types which can be used as parameter for the query_messages operation.

### 9.2.10 PLM_exception classes

All operations of the interfaces of the Computational Viewpoint can raise exceptions derived from PLM_exception. The following subtypes of PLM_exception the following exceptions are defined in this specification: Authentification_exception, Authorization_exception, Session_timeout_exception, Invalid_session_id_exception, Unsupported_query_exception, Unsupported_operation_exception, Object_uid_timeout_exception, and Invalid_object_uid_exception.

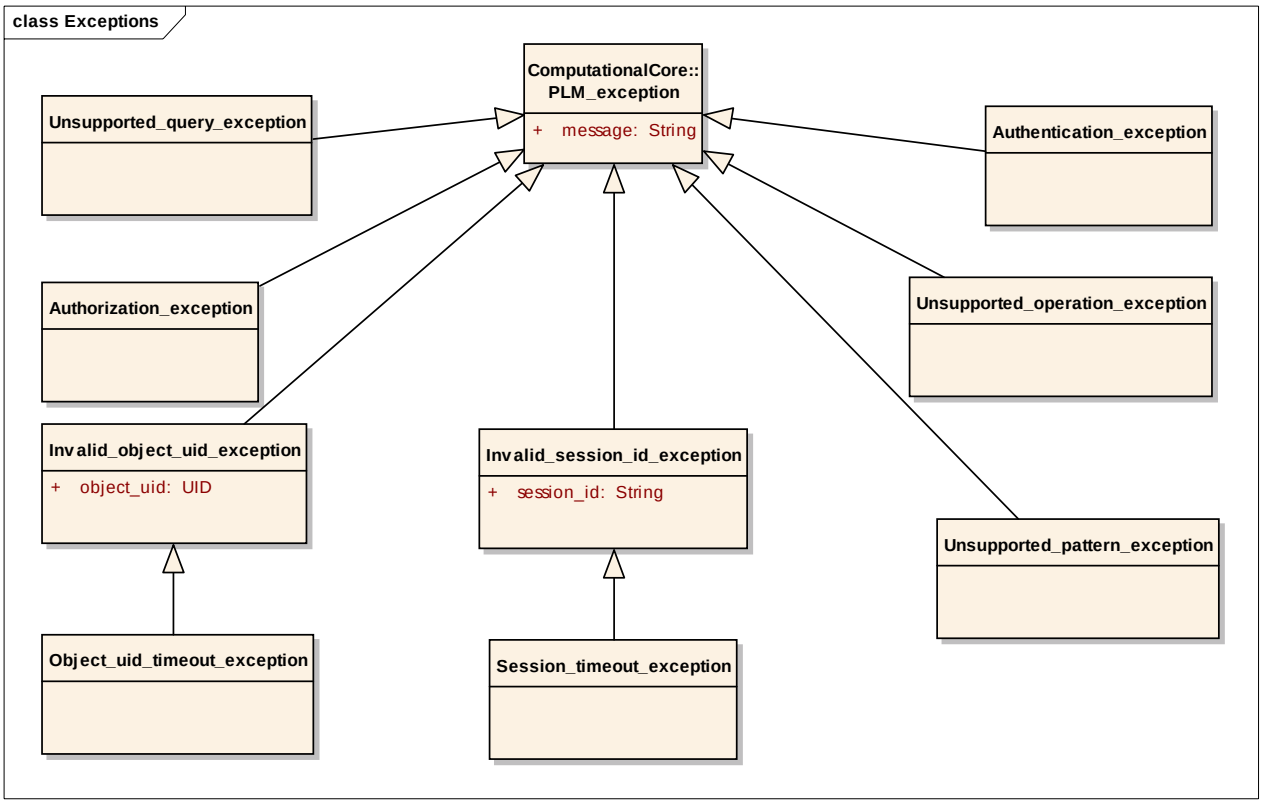


**Figure 9.10 - PLM_exception and its subtypes**

#### 9.2.10.1 Authentification_exception

The Authentification_exception is thrown by the operation get_connection of the interface PLM_connection_factory when the authentication of the client fails. The authentication mechanism is implementation specific.

#### 9.2.10.2 Authorization_exception

The Authorization_exception is thrown by arbitrary operations if the client has not the right to perform the requested operation with the given parameters.

#### 9.2.10.3 Invalid_object_uid_exception

The Invalid_object_uid_exception is thrown by arbitrary operations of the interface PLM_connection when a UID value of a server object is used in one parameter of the operation the associated object of which no longer exist or had never exist on the server. The UID value is returned in the attribute object_uid of the exception.

#### 9.2.10.4 Invalid_session_id_exception

The Invalid_session_id_exception is thrown by arbitrary operations of the interfaces PLM_general_connection and PLM_message_connection if a session identifier is used for that operation which is unknown to the service implementation. The transfer of session identifiers has to be defined by the platform specific models.

#### 9.2.10.5 Object_uid_timeout_exception

It is allowed to an implementation to end the validity time of an object UID before a session is closed. The Object_uid_timeout_exception must be thrown by an operation of the interface PLM_connection if such an object UID is used by a client in a parameter.

#### 9.2.10.6 Session_timeout_exception

The Session_timeout_exception is thrown by arbitrary operations of the interface PLM_connection when the session time has expired.

#### 9.2.10.7 Unsupported_query_exception

The Unsupported_query_exception is thrown by the query and export_data operation of the interface PLM_general_connection and by the query_messages operation of the interface PLM_message_connection if a PLM_query or PLM_message_query value is used as parameter, that is not supported by the service implementation.

#### 9.2.10.8 Unsupported_pattern_exception

The Unsupported_pattern_exception is thrown by the query() and export_data() operations of the interface PLM_general_connection and by the query_messages operation of the interface PLM_message_connection if in the parameter a pattern value is used which is not supported by the service implementation.

#### 9.2.10.9 Unsupported_operation_exception

The Unsupported_operation_exception is thrown by arbitrary operations of arbitrary interfaces when the requested operation is not supported by an service implementation.

### 9.2.11 PLM_query Type

The type Query is an abstract base type. It is used as parameter in the query and export_data operation of the PLM_general_connection. The type PLM_query is specialized in “Queries Conformance Points” Chapter 9.3.

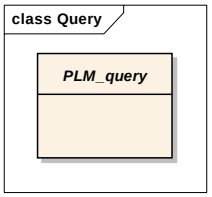


**Figure 9.11 - - PLM_query Type**

When a PLM_query instance is applied to the set of PLM_objects of a server it selects a subset of these PLM_objects. The way of selecting this initial result set is specific to each specialization of the PLM_query type. The initial result set of the PLM_query instance has to be extended by further PLM_objects of the server until the minimal result set is selected, which contains all initially selected PLM_objects and fulfills all occurrency constraints of all selected PLM_objects. This specification defines the following rules how to extend an initial result set to fulfill the multiplicity constaints:

- If a selected PLM_object instance is a component in a composition, the result set has to be extended by the composite instance.
- If a selected PLM_object instance is a composite in a composition and the multiplicity of the component in the composition is one, the result set has to be extended by the component instance.
- If a selected PLM_object instance has a reference and the multiplicity of the referenced objects is one, the result set has to be extended by the referenced instance.
- If a selected PLM_object instance is a composite in a composition or has a reference and the minimum multiplicity of the component respectively referenced objects is not zero and the maximum multiplicity is greater than one the result set is not extended by selecting further PLM_object instances. Either there are enough PLM_object instances selected in the result set that play the component role in the composition respectively the referenced role in the directed association to fulfill the minimum multilplicity constraint or the result set is extended by NIL objects which are used as components respectively referenced objects. NIL objects are special instances of types derived from PLM_object which can be used as helper instances to fulfill multiplicity constraints in PLM_object sets. The creation and distinction of NIL objects has to be defined in platform specific models.

## 9.3 Utilities Queries Conformance Point

The Utilities Queries Conformance Point provides specializations of the PLM_query interface with the possibility of concatenated, recursive and batch queries. All queries defined in the Utilities Queries Conformance Point are directly or indirectly derived from the abstract base type Utility_query.

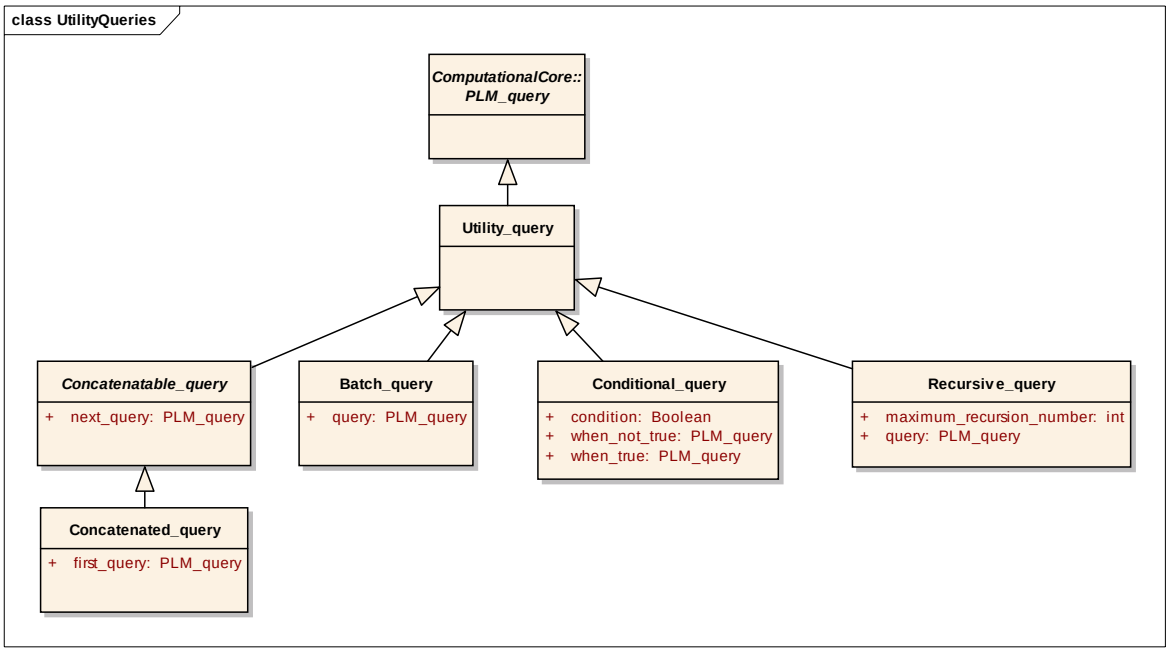


**Figure 9.12 - The Utility queries class diagram**

### 9.3.1 Concatenatable_query

The Concatenatable_query is a base interface for query types that provides the possibility concatenate other queries. The concatenation of queries is realized by an association which links a Concatenatable_query object with a next PLM_query object(s). The role name of the linked next PLM_query object(s) is next_query. If a query is extended by another query to a concatenated query, the result of the concatenated query is defined as the union of the results of the two single queries. The start nodes of the second query are limited to the nodes which the PLM_general_connection would return as result of the first query alone.

This limitation concerns only the start nodes but not the result of the second query. In the second query all links from the result nodes of the first query to arbitrary nodes in the PLM system can be evaluated and added to the result of the second query.

### 9.3.2 Concatenated_query

The Concatenated_query allows to use PLM_query instances as first query in a concatenation of queries which not implement the Concatenatable interface.

### 9.3.3 Recursive_query

The Recursive_query provides the possibility to execute other PLM_query instances recursively. In general, executing queries against a tree of PLM objects as defined by the Informational viewpoint would require a recursive tree traversal. This recursion query instance is controlled by Recursive_query that contains the PLM_query instance as its attribute query. The recursion is controlled by the attribute maximum_recursion_number of the containing Recursive_query. If this attribute is not set or has the value 0 the contained query is nonrecursive executed. If the attribute has a positive value n the contained query instance has n recursions. A recursion of a PLM_query instance has the same semantics as the concatenation of n equal PLM_query instances. A maximum_recursion_number with a negative value means an infinitive recursion.

### 9.3.4 Batch_query

The type Batch_query can combine other query instances to a batch job. A Batch_query instance is evaluated by evaluating all contained PLM_query instances of the Batch_query instance independently and create one result from all objects selected by the contained queries.

### 9.3.5 Conditional_query

The type Conditional_query enables the execution of a query in dependency of a condition. If the attribute condition is true the query referenced by when_true is executed otherwise the query referenced by when_not_true is executed.

## 9.4 Generic Queries Conformance Point

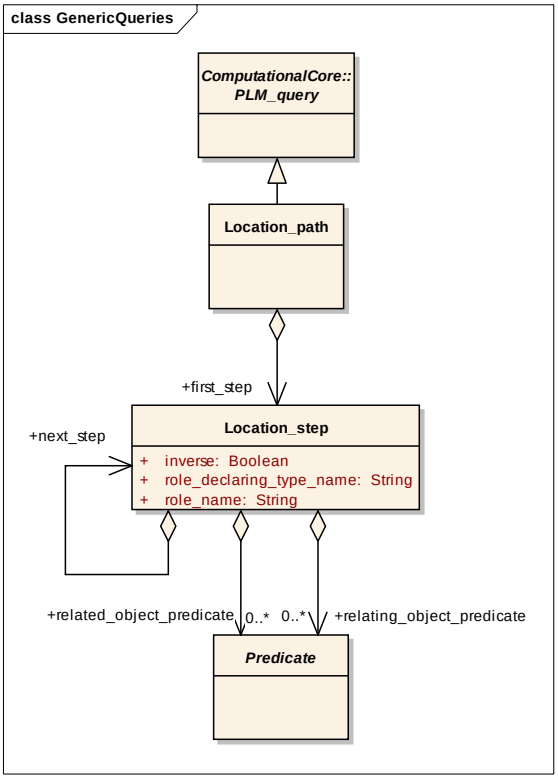


**Figure 9.13 - The class diagram of the Generic Queries Conformance Point**

The Generic Queries Conformance Point defines a toolset of classes that can be used to query arbitrary data from a PLM system. This toolset consists of the types Location_path, Location_step, Predicate and specializations of Predicate.The behaviour of concatenation, batch, recursive and condition processing is captured in specializations of this query type which is defined in a newly introduced Utilities Queries Conformance Point.

The PLM_container instance models PLM data as a set of direct or indirect contained nodes (instances of PLM_Object). The nodes are related by relationships. The relationship types of a node are composition or directed association. They are described in Section 8 for each node type.

To define a subset of the nodes of a PLM_container instance an instance of the abstract type Query has to be used. The type Location_path is the specialization of the Query type for the Generic Conformance Point. The Location_path is an new query tool that is de-signed to optimally implement the PLM Services needs. A Location_path consists of a tree of instances of Location_step.

The root node of the Tree is defined by the association first_step of the Location_path. By the association next_step of a Location_step instance the child nodes of this Location_step instance node in the tree are determined.

By applying a Location_path instance to a PLM_container instance each Location_step of the path in turn selects a set of nodes relative to the currently selected node-set.

The initially selected node-set is defined by all nodes that are directly or indirectly related to the PLM_container instance. The resulting selected node-set of a Location_path is the union of all selected node-sets of all Location_steps of the Location_path.

A location step consists of:

- a role name which specifies the nodes selected by the location step,
- the name of the type that declares the relationship with the role,
- a flag that indicates if the navigation direction is inverse in respect of the informational model,
- zero or more predicates which use arbitrary expressions further refining the set of start nodes selected by the location step,
- zero or more predicates which use arbitrary expressions further refining the set of nodes selected by the location step, and
- a list of location steps following directly the current location step.

The node-set selected by a location step is the node-set that results from generating an initial node-set from all nodes that are reached from the nodes in the current selected node-set by following the named relationship, and then filtering that node-set by each of the predicates in turn. If a Location_step has more than one next_step these steps results in one different selected node-set for each step.

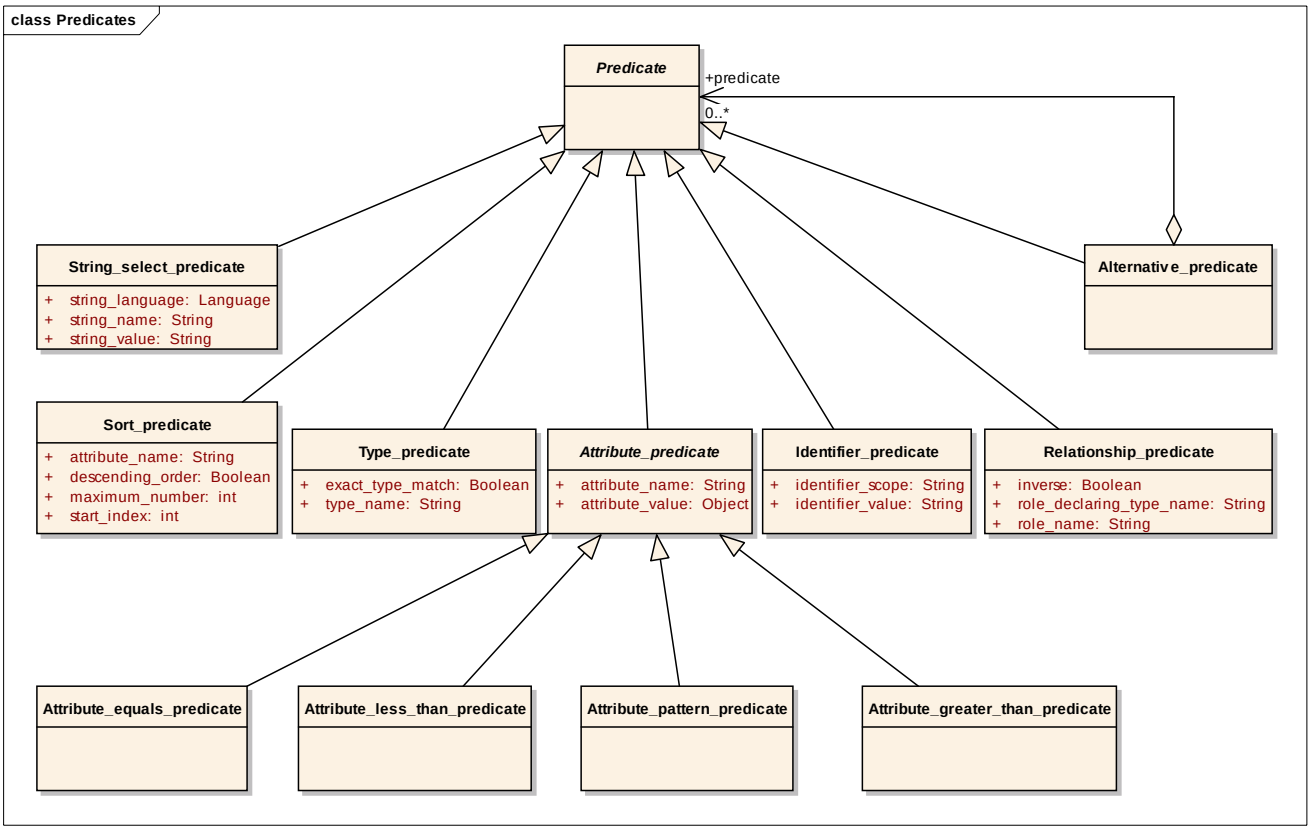


**Figure 9.14 - Class diagram of predicates and attributes**

Each non-abstract spezialisation of the abstract class Predicate defines a constraint for filtering object sets. Filtering means that the algorithm is applied to each object in the set and only the objects which fit the constraint remain in the set. The following non-abstract specializations of the class Predicate are defined in this specification:

### 9.4.1 Alternative_predicate

An object fulfill an Alternative_predicate constraint if it fulfill at least one the Predicate instances referenced by the relationship predicate of the Alternative_predicate.

### 9.4.2 Attribute_equals_predicate

An object fulfill an Attribute_equals_predicate if it has an attribute with the name given in the attribute attribute_name of the Attribute_equals_predicate and if that attribute has a value which is equals to the value given by the attribute attribute_value of the Attribute_equals_predicate.

### 9.4.3 Attribute_greater_than_predicate

An object fulfill an Attribute_greater_than_predicate if it has an attribute with the name given in the attribute attribute_name of the Attribute_greater_than_predicate and if that attribute has a value which is greater than the value given by the attribute attribute_value of the Attribute_greater_than_predicate.

### 9.4.4 Attribute_less_than_predicate

An object fulfill an Attribute_less_than_predicate if it has an attribute with the name given in the attribute attribute_name of the Attribute_less_than_predicate and if that attribute has a value which is less than the value given by the attribute attribute_value of the Attribute_less_than_predicate.

### 9.4.5 Attribute_pattern_predicate

An object fulfill an Attribute_pattern_predicate if it has an attribute with the name given in the attribute attribute_name of the Attribute_pattern_predicate and if that attribute has a value which match the pattern given by the attribute attribute_value of the Attribute_pattern_predicate. This specification uses the pattern language defined in [XML Schema W3C Recommendation 28 October 2004].

### 9.4.6 Identifier_predicate

All classes which have a composition of type Alias_identification have also an attribute that corresponds with the attribute alias_id of the related Alias_identification. These corresponding attributes are identifying attributes and can be filtered by Identifier_predicates.

There are three variants how an object can fulfill the constraints of an Identifier_predicate.

- If the attribute identifier_scope of the Identifier_predicate is not set, an object fulfill the Identifier_predicate if it has an identifier attribute and if that attribute has a value that matches the pattern given by the attribute identifier_value of the Identifier_predicate.
- If the attribute identifier_scope of an Identifier_predicate is set, an object fulfill the Identifier_predicate if it has an Alias_identification with an value for its attribute alias_scope that is equals to the value of the attribute identifier_scope and if the attribute alias_id of the Alias_identification has a value that matches the pattern given by the attribute identifier_value of the Identifier_predicate.
- If the attribute identifier_scope of an Identifier_predicate is set, an object fulfill the Identifier_predicate if it has an identifier attribute and if that attribute has a value that matches the pattern given by the attribute identifier_value of the Identifier_predicate and if it is referenced by the relationship is_applied_to of a Person_organization_assignment_instance and the attribute role of the Person_organization_assignment instance has the value "id owner" and the Person_organization_assignment is referenced by the composition person_organization_assignment of an Organization instance and the attribute id of the Organization instance is equals to the value of the attribute identifier_scope of the Identifier_predicate.

This specification uses the pattern language defined in [XML Schema W3C Recommendation 28 October 2004].

### 9.4.7 Relationship_predicate

An object fulfill a Relationship_predicate constraint if it fulfill the following partial constraints:

- The object is related with another object that fulfill all the Predicate instances referenced by the relationship predicate of the Relationship_predicate.
- If the value of the attribute inverse of the Relationship_predicate is not true and if the attribute role_name is set, the role name of the other object in the relationship must be equals to the value of the attribute role_name of the Relationship_predicate.

- If the value of the attribute inverse of the Relationship_predicate is true and if the attribute role_name is set, the role name of this object in the relationship must be equals to the value of the attribute role_name of the Relationship_predicate.
- If the attribute role_declaring_type_name is set, the relationship must be defined in a type which name is equals to the value of the attribute role_decalring_type_name.

### 9.4.8 String_select_predicate

An object fulfill a String_select_predicate if it has an attribute of type String_select with the name given in the attribute string_name of the String_select_predicate and if it fulfill one of the following constaints:

- If the attribute string_language is not set and the attribute with the name given by the attribute string_name must be a Default_language_string which value is equals to the value given by the attribute string_value of the String_select_predicate.
- If the attribute string_language is set and equals the default language of the server implementation the attribute with the name given by the attribute string_name must be an instance of Default_language_string_with a value which is equals to the value given by the attribute string_value of the String_select_predicate or an instance of Multi_language_string with a primary_language_dependent_string which value is equals to the value given by the attribute string_value of the String_select_predicate.
- If the attribute string_language is set and not equals the default language of the server implementation the attribute with the name given by the attribute string_name must be an instanceof Multi_language_string and have a additional_language_dependent_string which value is equals to the value given by the attribute string_value of the String_select_predicate.

### 9.4.9 Type_predicate

If the value of the attribute exact_type_match of an Type_predicate is TRUE, an object fulfill that Type_predicate constraint if it has exact the type specified in the attribute type_name of the Type_predicate.

If the value of the attribute exact_type_match of an Type_predicate is not TRUE, an object fulfill that Type_predicate constraint if it is an instance of the type specified in the attribute type_name of the Type_predicate or an instance of a derivation of that type.

### 9.4.10 Sort_predicate

The Sort_predicate sorts the objects selected by a Location_step by the values of the one attribute of those objects specified by the attribute attribute_name oth the Sort_predicate. The type of the attribute specified by the attribute_name must be a type for which a greater-lesser-relation is defined. The default sort order is ascending order. If the attribute Descending_order of the Sort_predicate has a value of TRUE the selected objects are sorted in descending order. If the attribute Start_index is set, from the sorted selected objects the objects before the start index are filtered out. The default start index is 0. If the attribute Maximum_number is set, from the sorted selected objects the objects after Start_index+ Maximum_number are filtered out.

## 9.5 XPath Queries Conformance Point

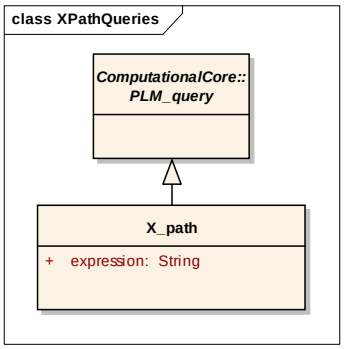


**Figure 9.15 - The class diagram of the XPath Queries Conformance Point**

The XPath conformance point defines the type X_path as specialization of the type Query. The type X_path provides the possibility to use arbitrary XPath expressions conforming to the W3C XPath specification as queries. The Web Service PSM defined in this specification defines how a PLM_container instance has to be transformed to a XML-Document. An XPath expression selects nodes in this XML-Document. These nodes (or their parent nodes in the case of non XML element nodes) have equivalent instances in the PIM that are subtypes of PLM_object. These instances are the result set of a XPath expression at the PIM level.

## 9.6 Proxy Queries Conformance Point

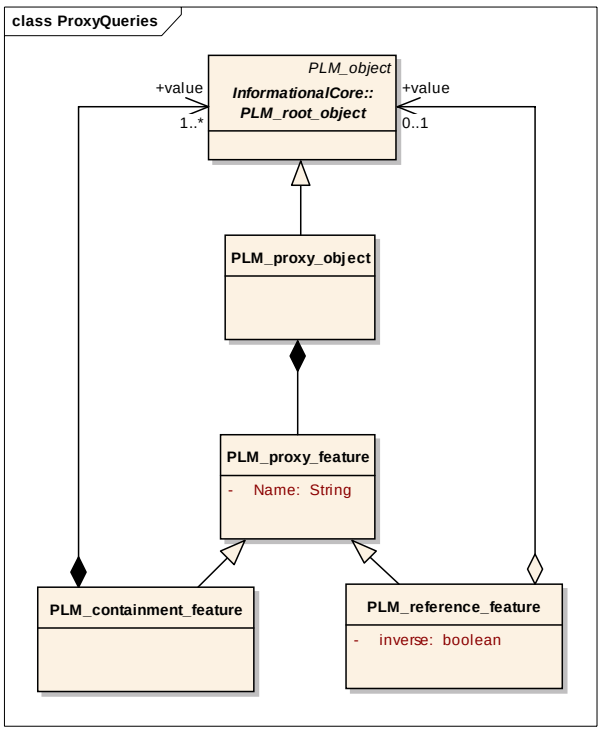


**Figure 9.16 - The class diagram of the Proxy Queries Conformance Point**

The Proxy Queries Conformance Point provides the possebility to avoid redundant data transfers in multiple query results and so to reduce the transferred data size. For that purpose a spezial container type is derived from the PLM_container interface that can contain beside complete PLM_objects so called proxy objects with reduced features. A client can ask proxy objects if it has stored the complete features of some objects in earlier requests.

### 9.6.1 PLM_proxy_object

The PLM_proxy_object is a proxy for an arbitrary object derived from PLM_object and identifies the PLM object by its uid. If a client wants to receive a proxy object instead a complete PLM object, it has to use the PLM_proxy_query instead of the Object_by_uid_query. PLM_proxy_objects can also be used in the container parameter of the write operation of the PLM_general_connection interface to update single features of PLM_objects.

#### *Base Class*

- PLM_root_object

#### *Compositions*

- PLM_proxy_feature: PLM_proxy_feature [0..*]

### 9.6.2 PLM_proxy_feature

The PLM_proxy_feature is the abstract base class for reference and containment features.

#### *Attributes*

- name: String

### 9.6.3 PLM_containment_feature

The PLM_containment_feature models a containment relation between a PLM_proxy_object (container) and one or more PLM_objects (containeds).

#### *Base Class*

- PLM_proxy_feature

#### *Compositions*

- value: PLM_object [1..*]

### 9.6.4 PLM_reference_feature

The PLM_reference_feature models a reference between a PLM_proxy_object and one or more PLM_objects. If the attribute inverse is TRUE the PLM_reference_feature models references from other PLM objects to the PLM object represented by the parent object of the PLM_reference_feature, otherwise it models references from the parent object of the PLM_reference_feature to other PLM objects.

#### *Base Class*

- PLM_proxy_feature

#### *References*

- value: PLM_object [1..*]

#### *Attributes*

- inverse: Boolean [0..1]

### 9.6.5 PLM_proxy_container

The PLM_proxy_container extends the PLM_container by the ability to contain PLM_proxy_objects.

#### *Base Class*

- PLM_container

#### *Compositions*

- PLM_propxy_object: PLM_proxy_object [0..*]

### 9.6.6 Proxy_query

The Proxy_query selects objects by its uid. The selected objects has to put into the result container not as copy but as PLM_proxy_objects. If next_queries of the Proxy_query traverses containments or references to other PLM_objects these containments or references has to be instantiated as PLM_containment_feature or PLM_reference_feature of the PLM_proxy_object. The client should provide not only the uid but also the opaque_server_data, of the objects it is interested in, if it has this data (e.g. from a former query result), because the service implementation can use this information for an optimized processing of the PLM_proxy_query.

#### *Base Class*

- Concatenatable_query

#### *Attributes*

- uid: UID
- opaque_server_data : Byte [0..*]

## 9.7 Specific Queries Conformance Point

The Specific Queries Conformance Point defines a set of low level specialized queries. The semantics of each specialized query of this conformance point is defined by an equivalent Location_path instance. The semantics of Location_path is defined in the Generic Queries Conformance Point.

### 9.7.1 Common Queries as base types for specific queries

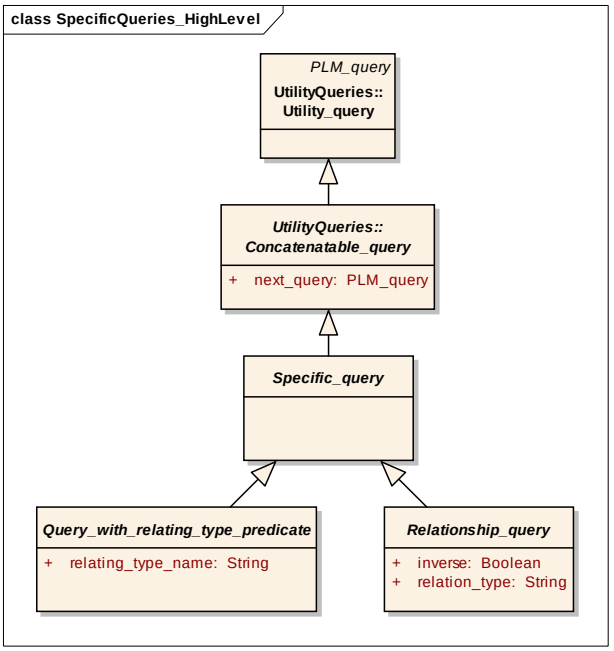


**Figure 9.17 - Common base types in the specific queries conformance point**

#### 9.7.1.1 Class Specific_query

All queries from the Specific Queries Point directly or indirectly derive from the Specific_query which implements the interface Concatenatable_query from the Utilities Queries Conformance Point.

#### 9.7.1.2 Class Query_with_relating_type_predicate

The abstract class Query_with_relating_type_predicate is used as base class for all queries which need an attribute relating_type_name.

#### 9.7.1.3 Class Relationship_query

The abstract class Relationship_query is used as base class for all queries which need an attribute relation_type and an attribute inverse.

### 9.7.2 Activity_ query

The Activity_authorization_query selects Activity objects.

***Base***

- Specific_query

***Parameters***

- Activity_type:string[0..1]
  filters Activity objects by their Activity_type attribute
- Id : string[0..1] filters Activity objects by their Id attribute
- Id_scope : string[0..1] filters Activity objects by the scope of their Id attribute
- Status : string[0..1] filters Activity objects by their Status attribute
- Resolved_request_id : string[0..1]
  filters Activity objects by the Id of their referenced Resolved_request object
- Concerned_organization_id : string[0..1]
  filters Activity objects by the Id of their referenced Concerned_organization objects
- Supplying_organization_id : string[0..1]
  filters Activity objects by the Id of their referenced Supplying_organization objects
- Requestor_id : string[0..1]
  filters Activity objects by the Id of their referenced Requestor object
- From_requestor_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Requestor date
- Till_requestor_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Requestor date
- From_actual_end_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Actual_end_date
- Till_actual_end_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Actual_end_date
- From_planned_end_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Planned_end_date
- Till_planned_end_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Planned_end_date
- From_actual_start_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Actual_start_date
- Till_actual_start_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Actual_start_date
- From_planned_start_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Planned_start_date
- Till_planned_start_date : dateTime[0..1]
  filters Activity objects by the Id of their referenced Planned_start_date
- Internal : boolean[0..1]
  filters Activity objects by the value of their Internal attribute

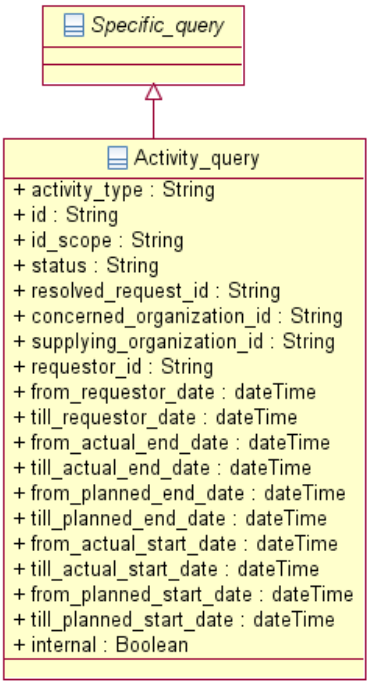


**Figure 9.18 - Definition of the Activity_query**

### 9.7.3 Activity_authorization_query

The Activity_authorization_query traverses from Activity objects to Work_order objects via the inverse Is_controlling relationship.

#### *Base Class*

- Specific_query

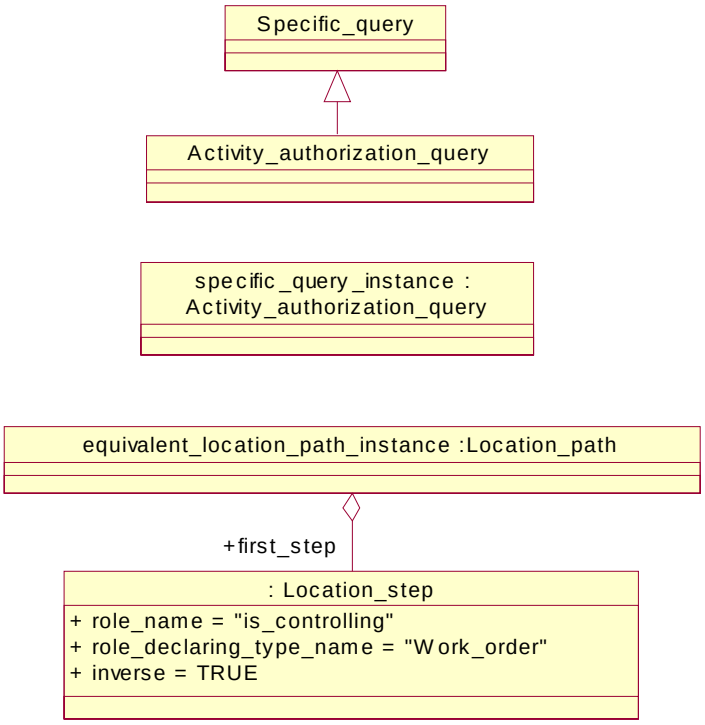


**Figure 9.19 - Definition, sample instance and equivalent Location_path instance of the Activity_authorization_query**

### 9.7.4 Activity_element_query

The Activity_element_query traverses from Activity objects via Activity_element objects to Activity_element_select objects.

#### *Base Class*

- Specific_query

#### *Parameters*

- role : String [0..1]
- element_type_name : String [0..1]

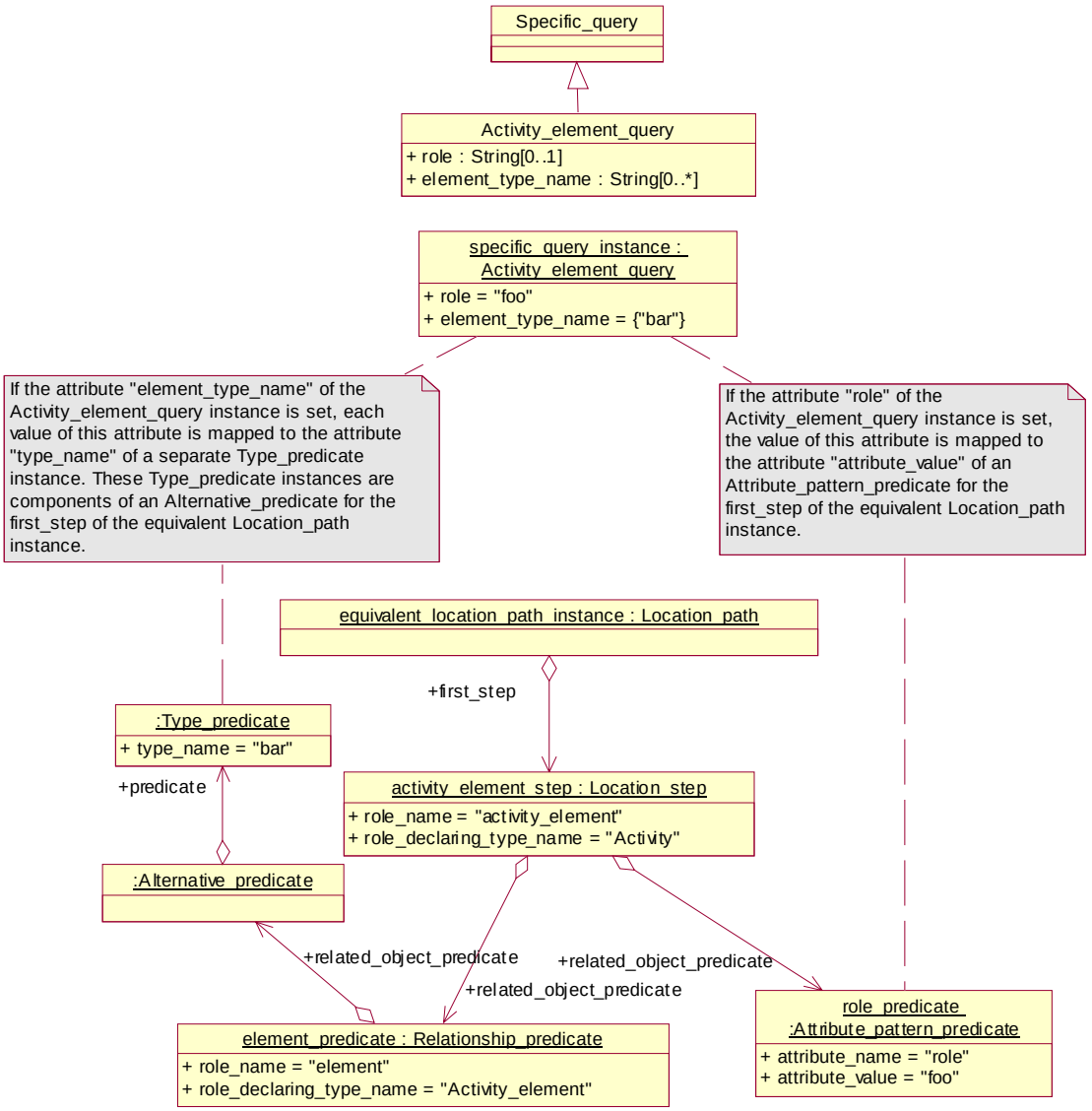


**Figure 9.20 - Definition, sample instance and equivalent Location_path instance of the Activity_element_query**

### 9.7.5 Activity_relationship_query

The Activity_relationship_query traverses from Activity objects via Activity_relationship objects to Activity objects.

#### *Base Class*

- Relationship_query

***Parameters***

- relation_type : String [0..1]

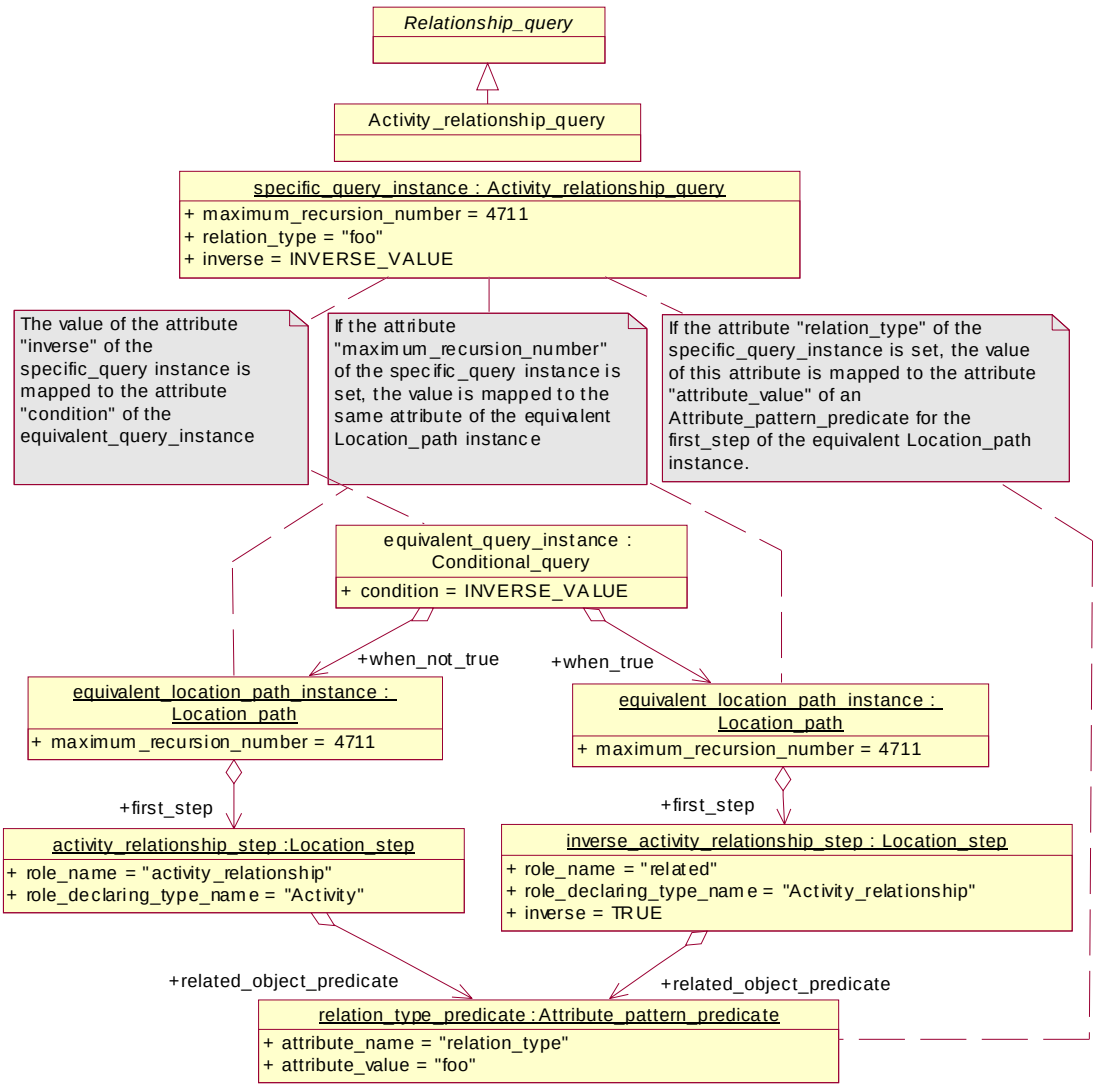


**Figure 9.21 - Definition, sample instance and equivalent Location_path instance of the Activity_relationship_query**

## 9.7.6 Activity_resolved_request_query

The Activity_resolved_request_query traverses from Activity objects to Work_request objects via the Resolved_request relationship.

***Base Class***

- Specific_query

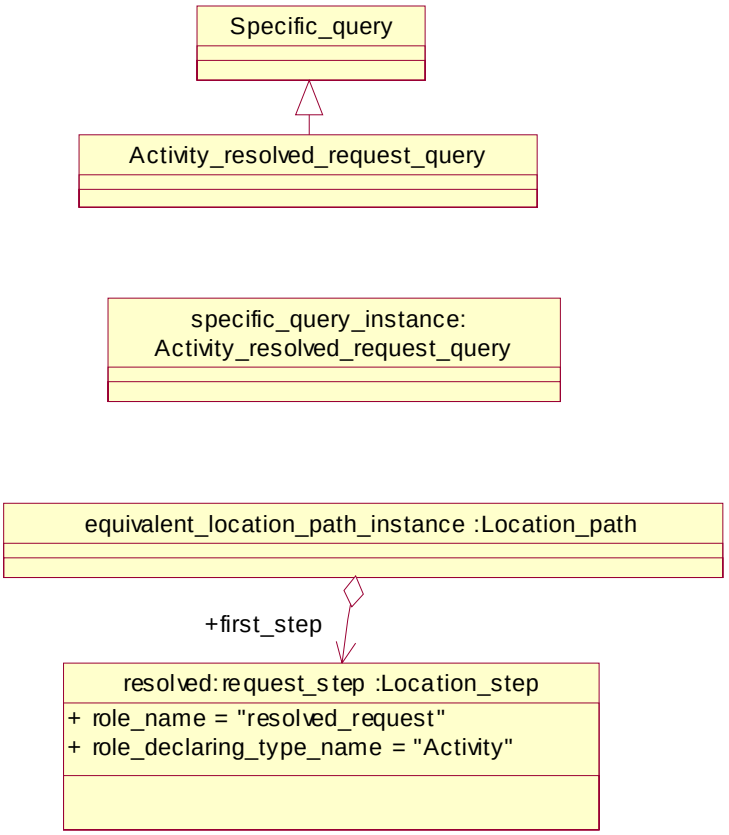


**Figure 9.22 - Definition, sample instance and equivalent Location_path instance of the Activity_resolved_request_query**

### 9.7.7 Alias_identification_query

The Alias_identification_query traverses alias information from instances which implements the interface Alias_select..

***Base Class***

- Query_with_releation_type_predicate

***Parameters***

- alias_id : String [0..1]
- alias_version_id : String [0..1]
- alias_scope : String [0..1]

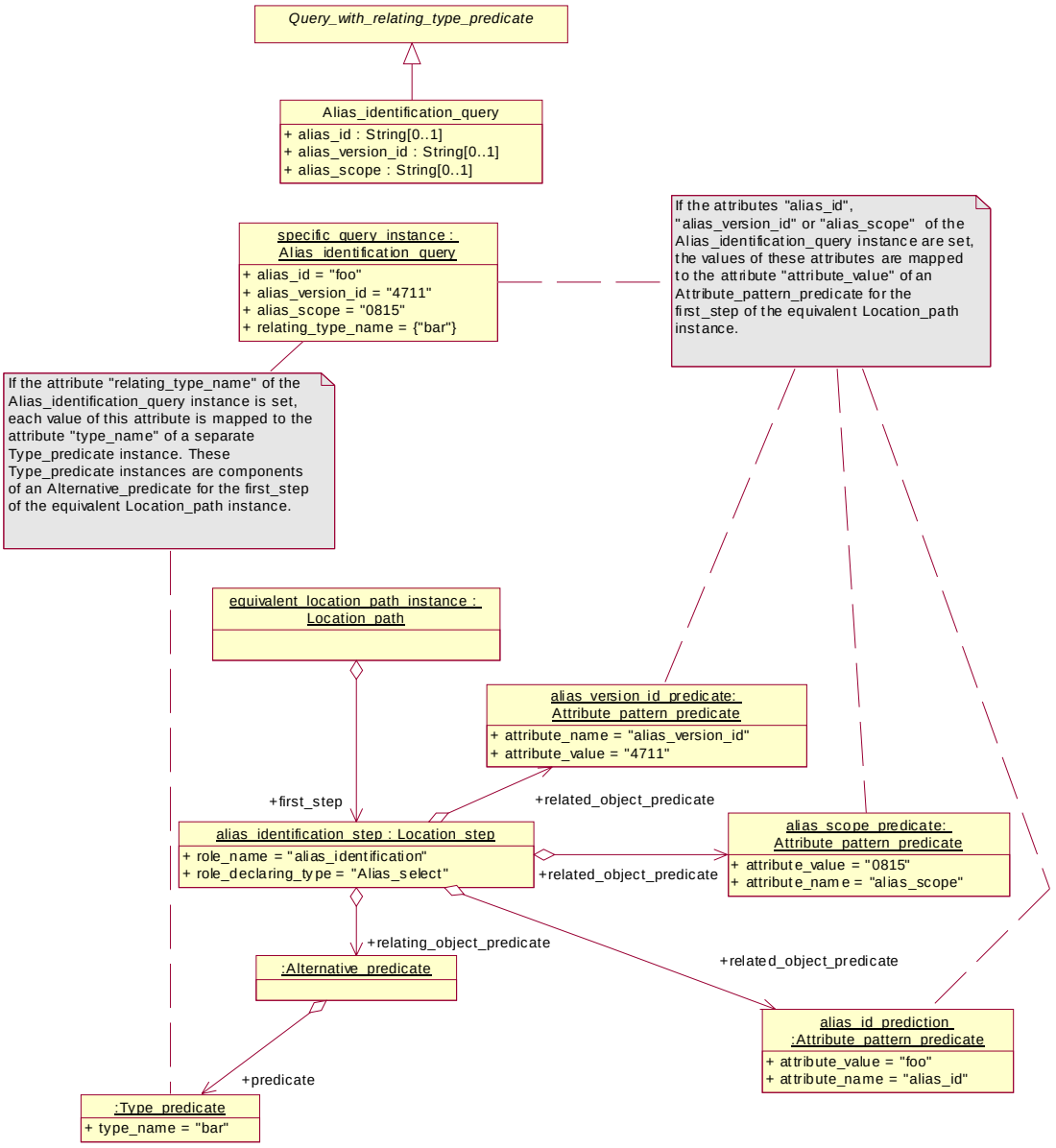


**Figure 9.23 - Definition, sample instance and equivalent Location_path instance of the Alias_identification_query**

### 9.7.8 Alternative_solution_query

The Alternative_solution_query traverses from Complex_product objects to Alternative_solution objects.

### *Base Class*

- Query_with_releation_type_predicate

### *Parameters*

- relating_type_name : String [0..*]

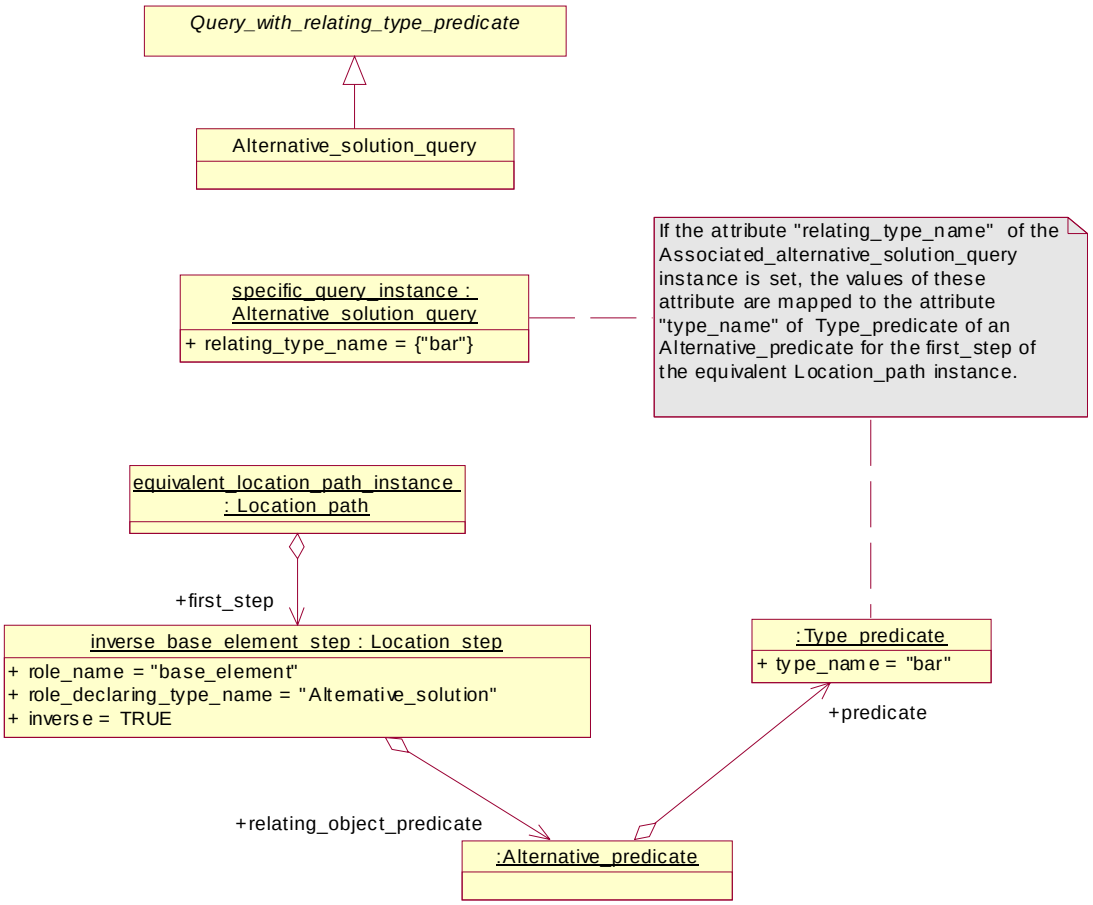


**Figure 9.24 - Definition, sample instance and equivalent Location_path instance of the Alternative_solution_query**

## 9.7.9 Application_context_query

The Application_context_query selects Application_context objects.

### *Base Class*

- Specific_query

### *Parameters*

- application_domain : String [0..1]
- life_cycle_stage : String [0..1

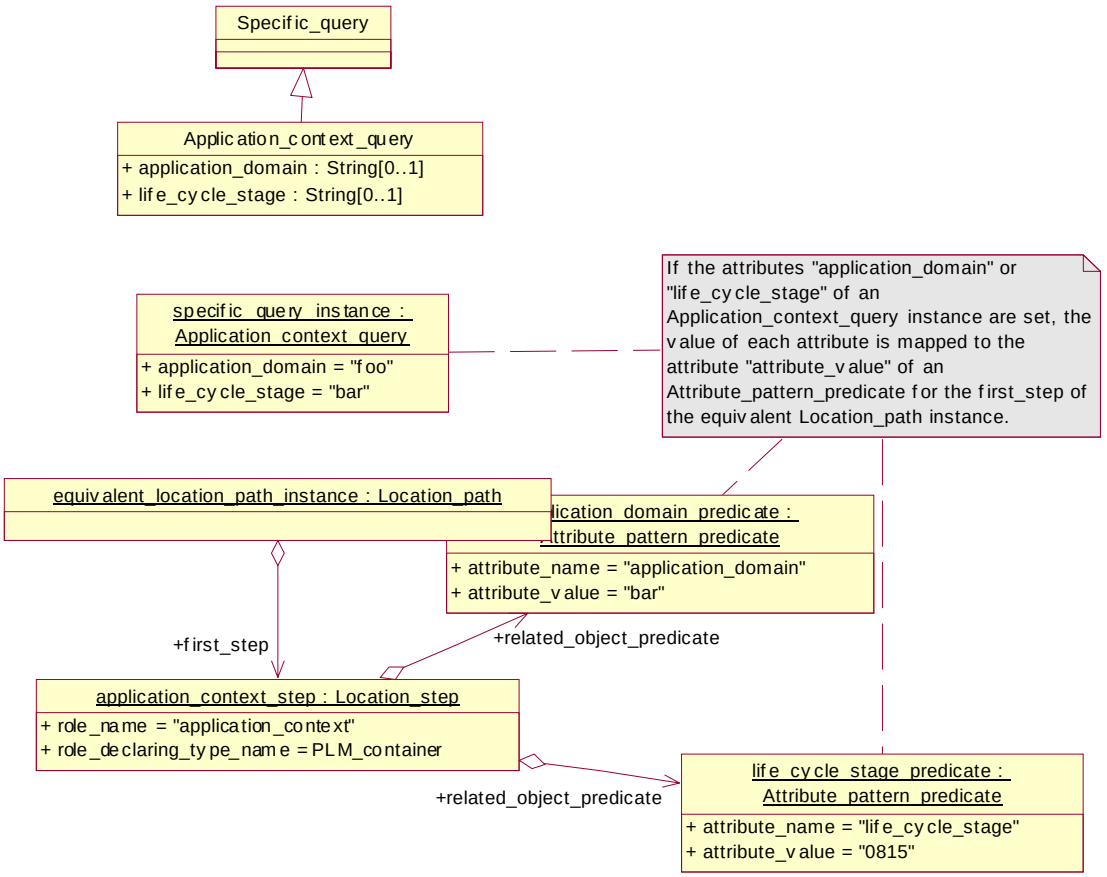


**Figure 9.25 - Definition, sample instance and equivalent Location_path instance of the Application_context_query**

## 9.7.10 Approval_relationship_query

The Approval_relationship_query traverses from Approval objects via Approval_relationship objects to Approval objects.

### *Base Class*

- Releationship_query

### *Parameters*

- relation_type : String [0..1]
- inverse : Boolean [0..1]

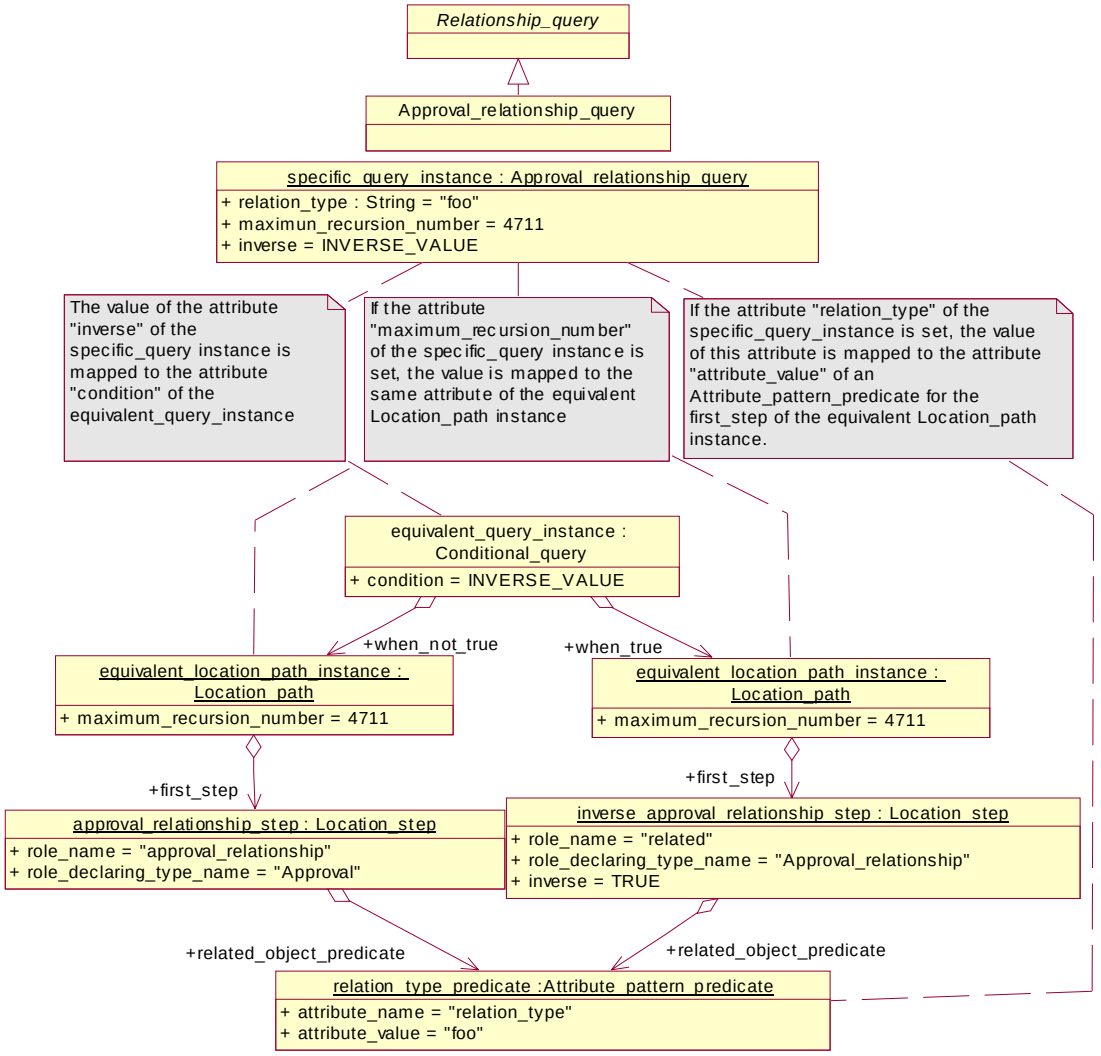


**Figure 9.26 - Definition, sample instance and equivalent Location_path instance of the Approval_relationship_query**

### 9.7.11 Assembly_component_placement_query

The Assembly_component_placement_query traverses from Assembly_component_relationship objects to Transformation_select objects.

#### *Base Class*

- Specific_query

#### *Parameters*

- none

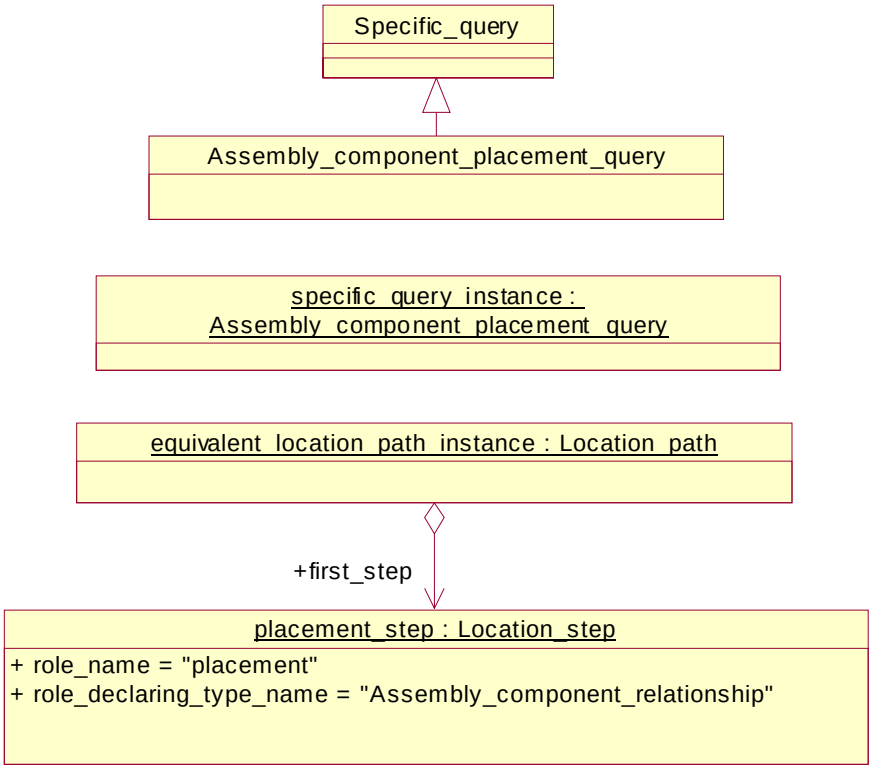


**Figure 9.27 - Definition, sample instance and equivalent Location_path instance of the Assembly_component_placememt_query**

### 9.7.12 Associated_activity_ query

The Associated_activity_query traverses from Activity_element_select objects via Activity_element objects to Activity objects.

#### *Base Class*

- Query_with_releation_type_predicate

#### *Parameters*

- relation_type : String [0..1]
- relating_type_name : String [0..1]

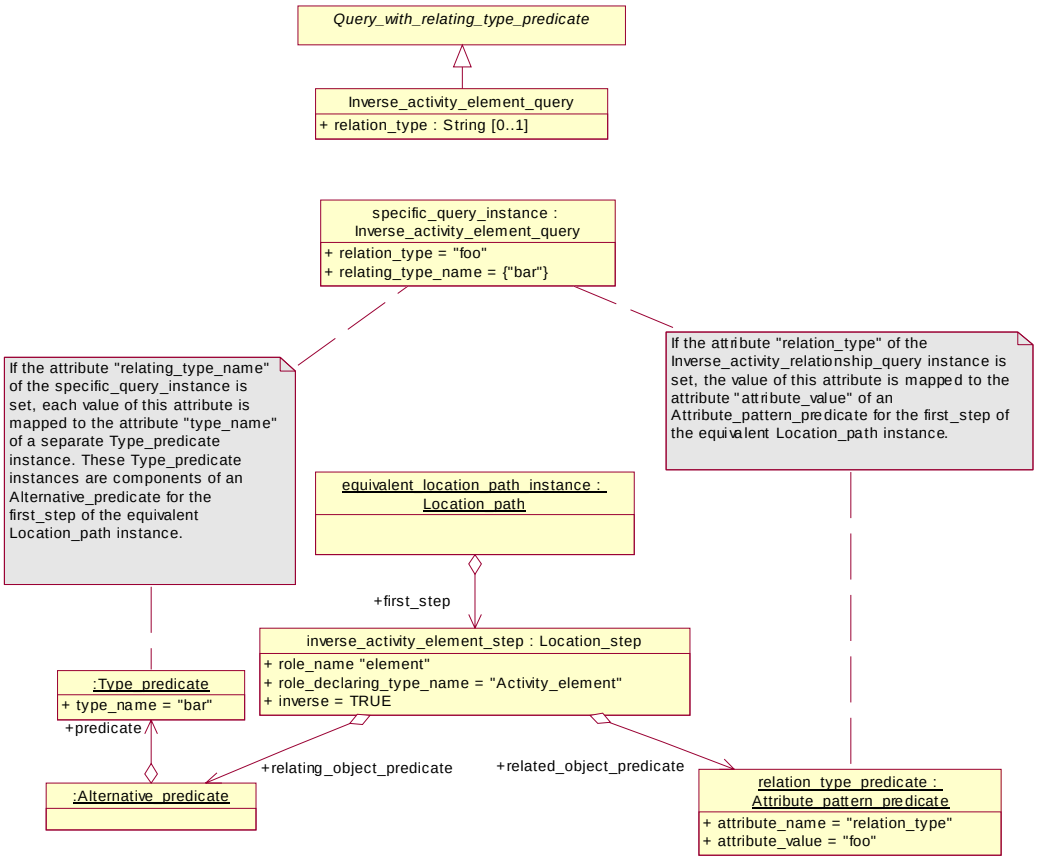


**Figure 9.28 - Definition, sample instance and equivalent Location_path instance of the Associated_activity_query**

## 9.7.13 Associated_approval_query

The Associated_approval_query traverses from Approval_element_select objects to Approval objects.

### *Base Class*

- Query_with_releation_type_predicate

### *Parameters*

- level : String [0..1]
- relating_type_name : String [0..*]

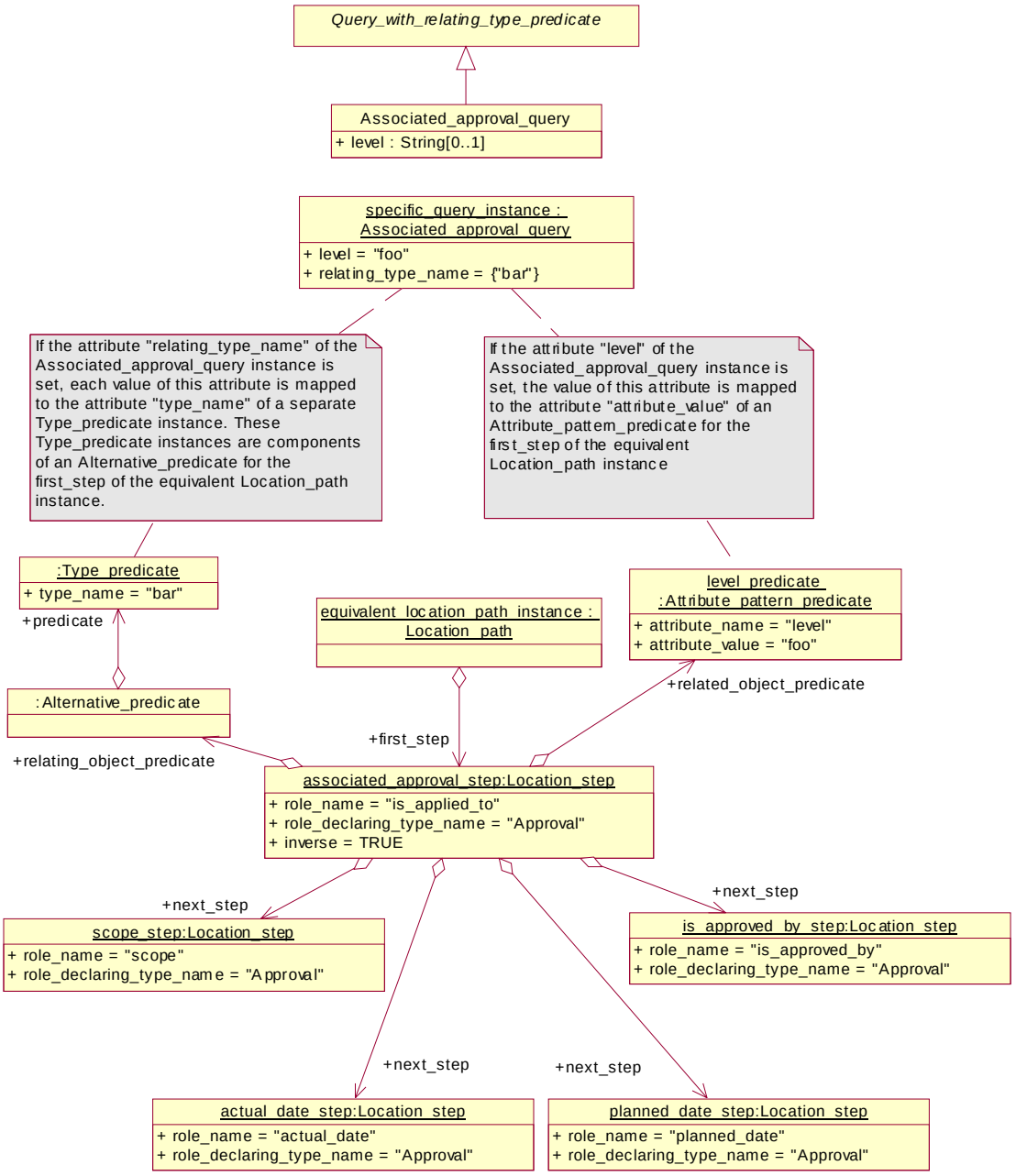


**Figure 9.29 - Definition, sample instance and equivalent Location_path instance of the Associated_approval_query**

### 9.7.14 Associated_classification_query

The Associated_classification_query traverses from Classified_element_select objects via Classification_association objects

to General_classification objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- role : String [0..1]
- relating_type_name : String [0..*]

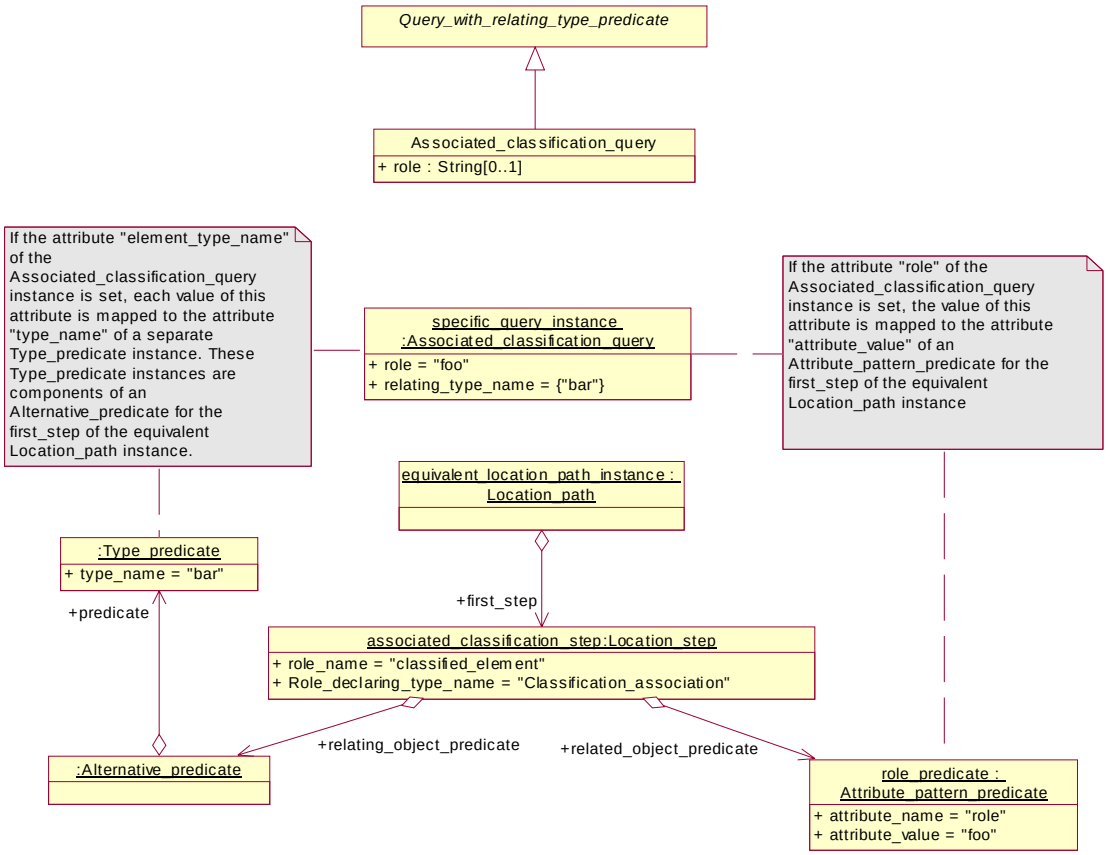


**Figure 9.30 - Definition, sample instance and equivalent Location_path instance of the Associated_classification_query**

## 9.7.15 Associated_date_organization_query

The Associated_date_organization_query traverses from Date_time_person_organization_element_select objects via Date_and_person_assignment objects to Date_and_person_organization objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- role : String [0..1]
- relating_type_name : String [0..*]

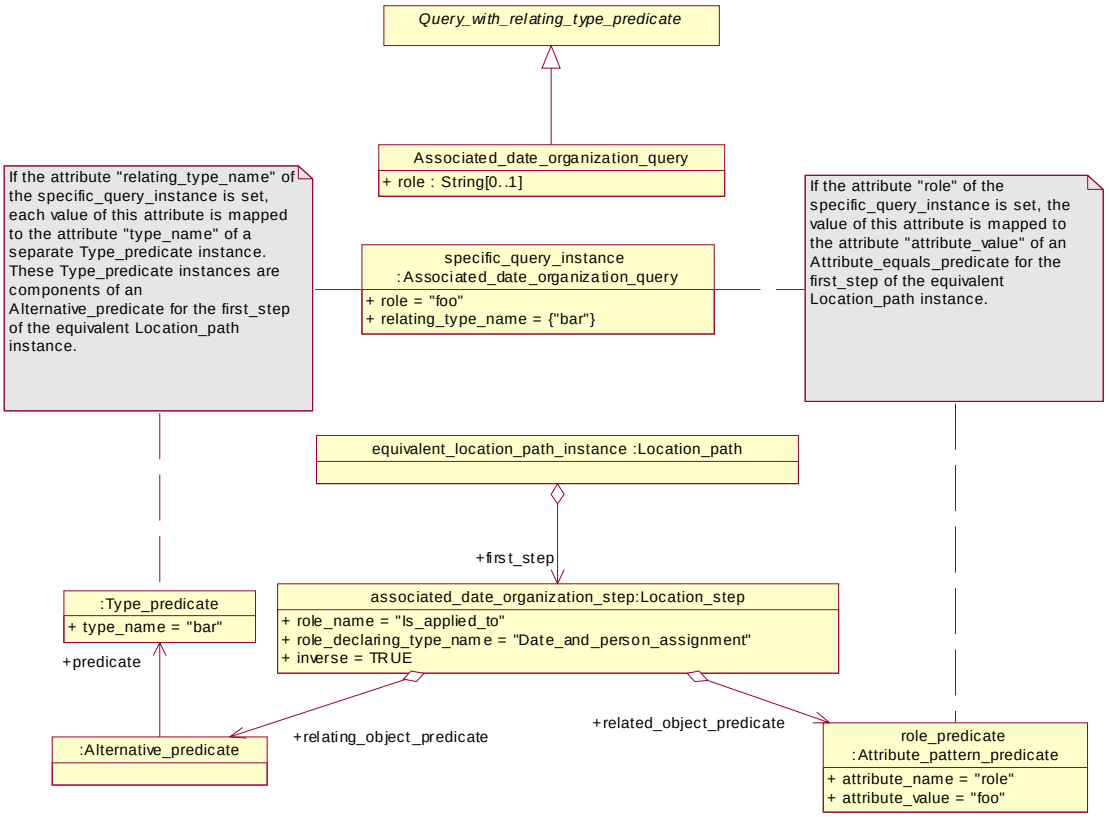


**Figure 9.31 - Definition, sample instance and equivalent Location_path instance of the Associated_date_organization_query**

## 9.7.16 Associated_date_time_query

The Associated_date_time_query traverses from Date_time_person_organization_element_select objects via Date_time_assignment objects to Date_time objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- role : String [0..1]
- relating_type_name : String [0..*]

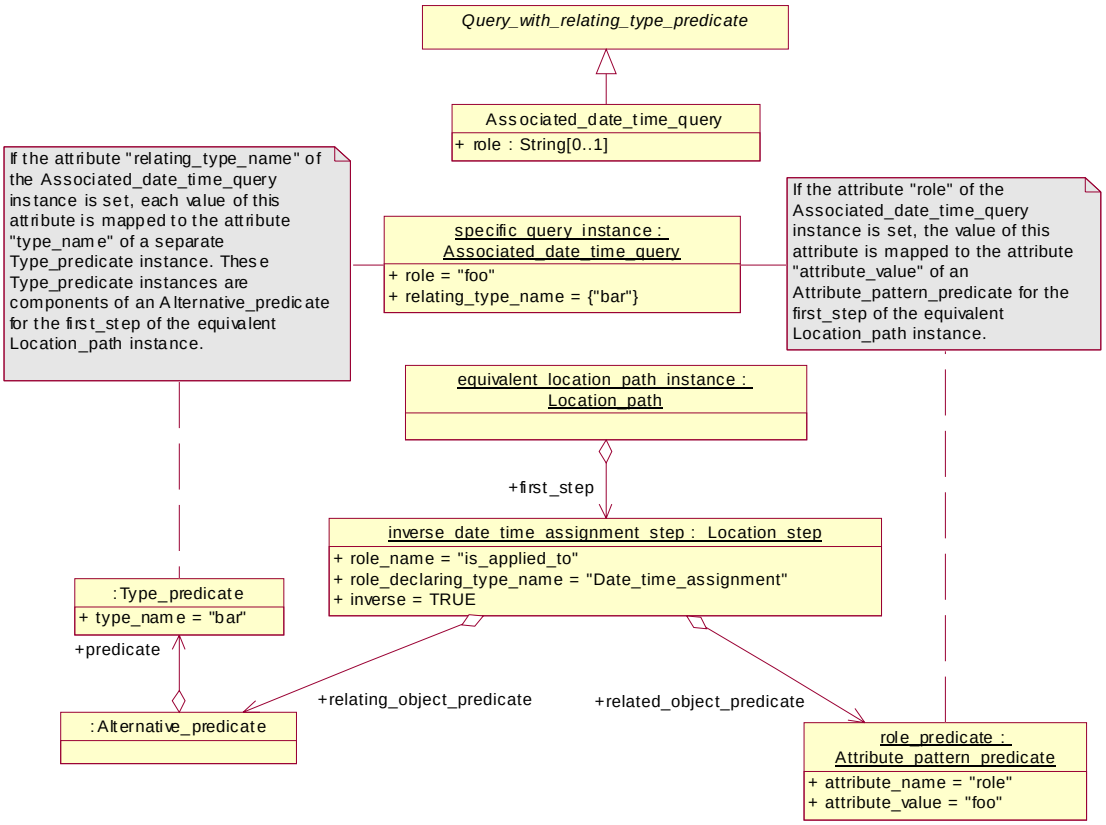


**Figure 9.32 - Definition, sample instance and equivalent Location_path instance of the Associated_date_time_query**

## 9.7.17 Associated_document_query

The Associated_document_query traverses from Documented_element_select objects via Document_assignment objects to Assigned_document_select objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- role : String [0..1]
- relating_type_name : String [0..*]
- include_file : Boolean [0..1]

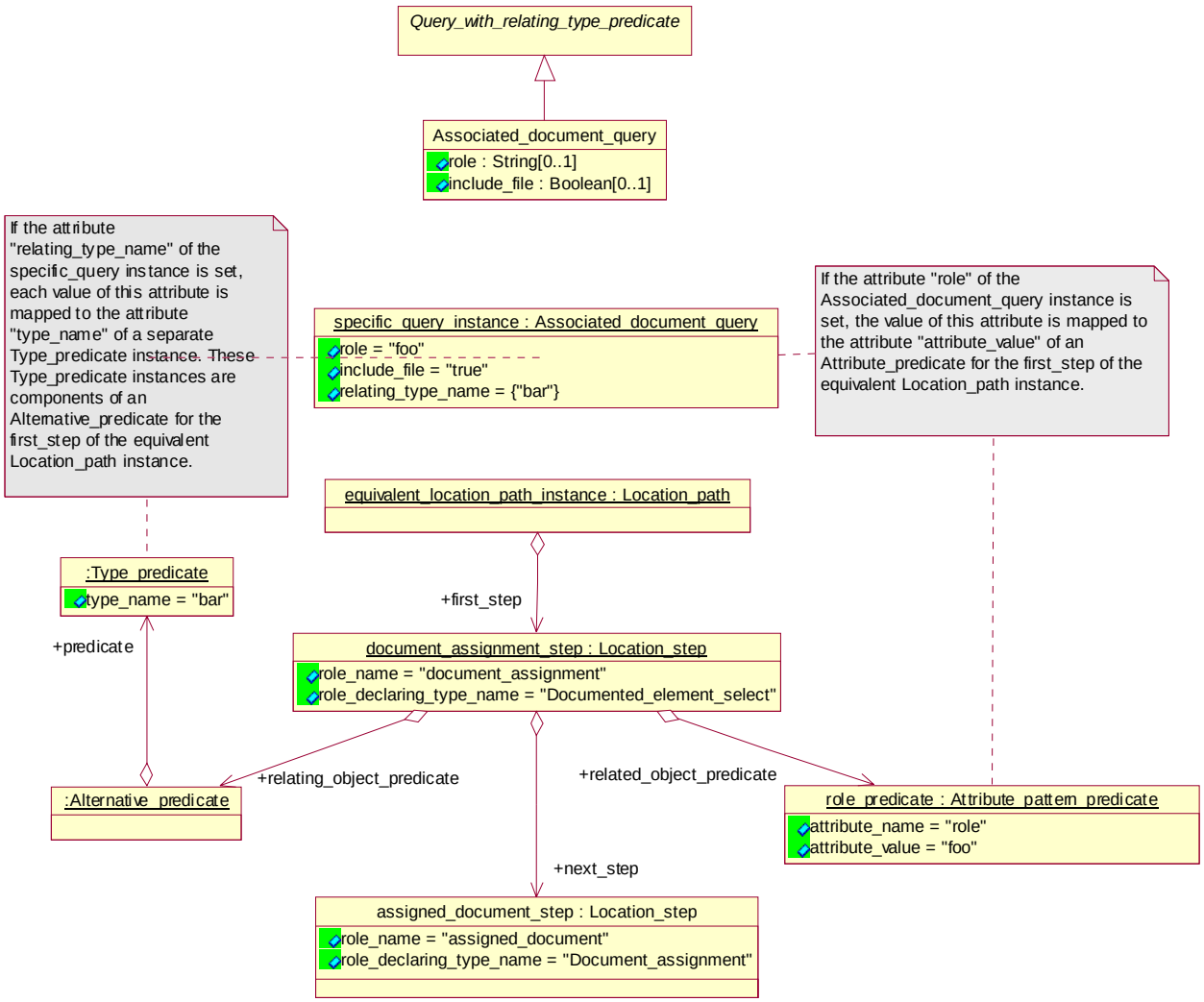


**Figure 9.33 - Definition, sample instance and equivalent Location_path instance of the Associated_document_query**

## 9.7.18 Associated_effectivity query

The Associated_effectivity_query traverses from Effective_element_select objects via Effectivity_assignment objects to Effectivity objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- role : String [0..1]

- relating_type_name : String [0..*]

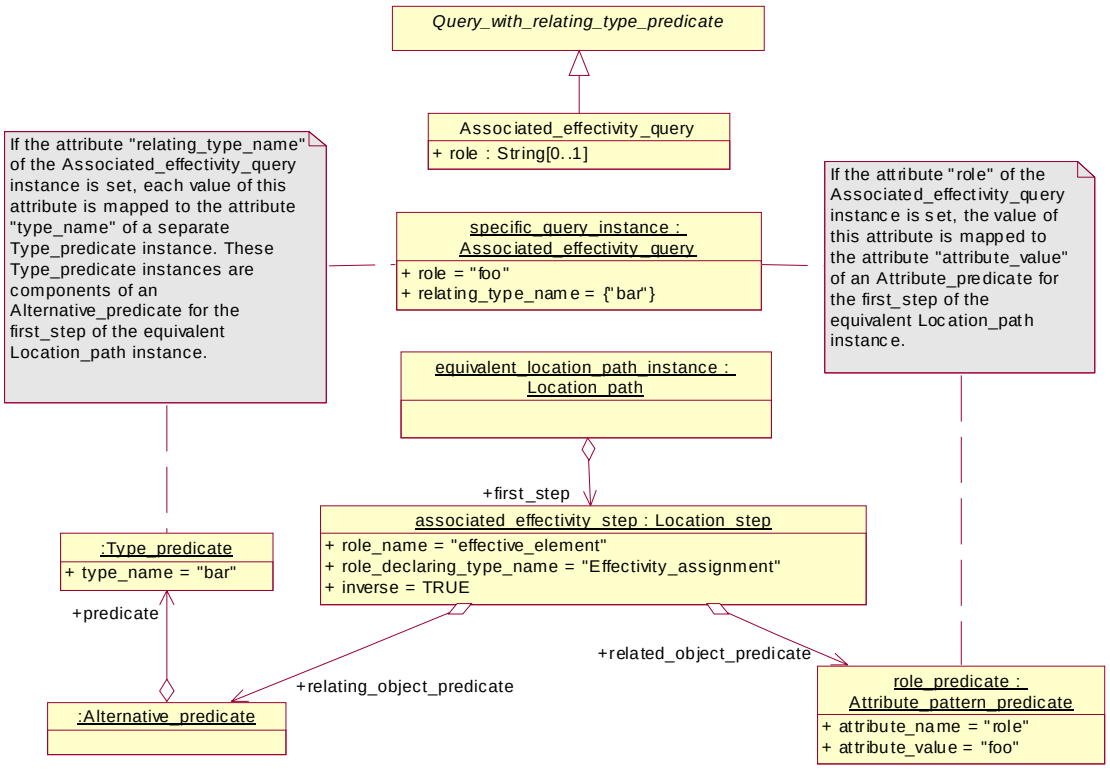


**Figure 9.34 - Definition, sample instance and equivalent Location_path instance of the Associated_effectivity _query**

### 9.7.19 Associated_file_query

The Associated_file_query traverses the Digital_file objects from Document_representation objects.

#### *Base Class*

- Query_with_relating_type_predicate

#### *Parameters*

- role : String [0..1]
- relating_type_name : String [0..*]
- include_file : Boolean [0..1]

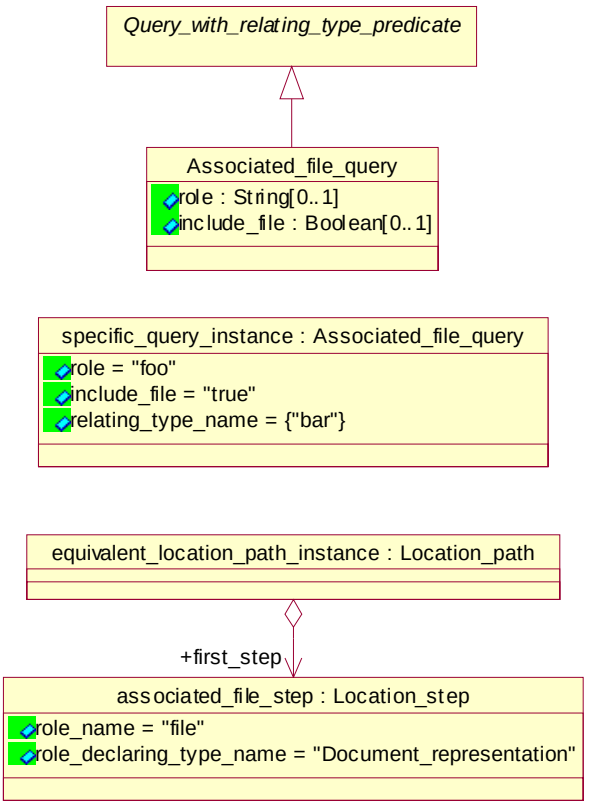


**Figure 9.35 - Definition, sample instance and equivalent Location_path instance of the Associated_file_query**

## 9.7.20 Associated_general_classification_query

The Associated_general_classification_query traverses from Classified_element_select objects to General_classification object via the Classification_association objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- Id: string [0..1]
- Id_scope: string [0..1]
- Version_id: string [0..1]
- Role: string [0..1]

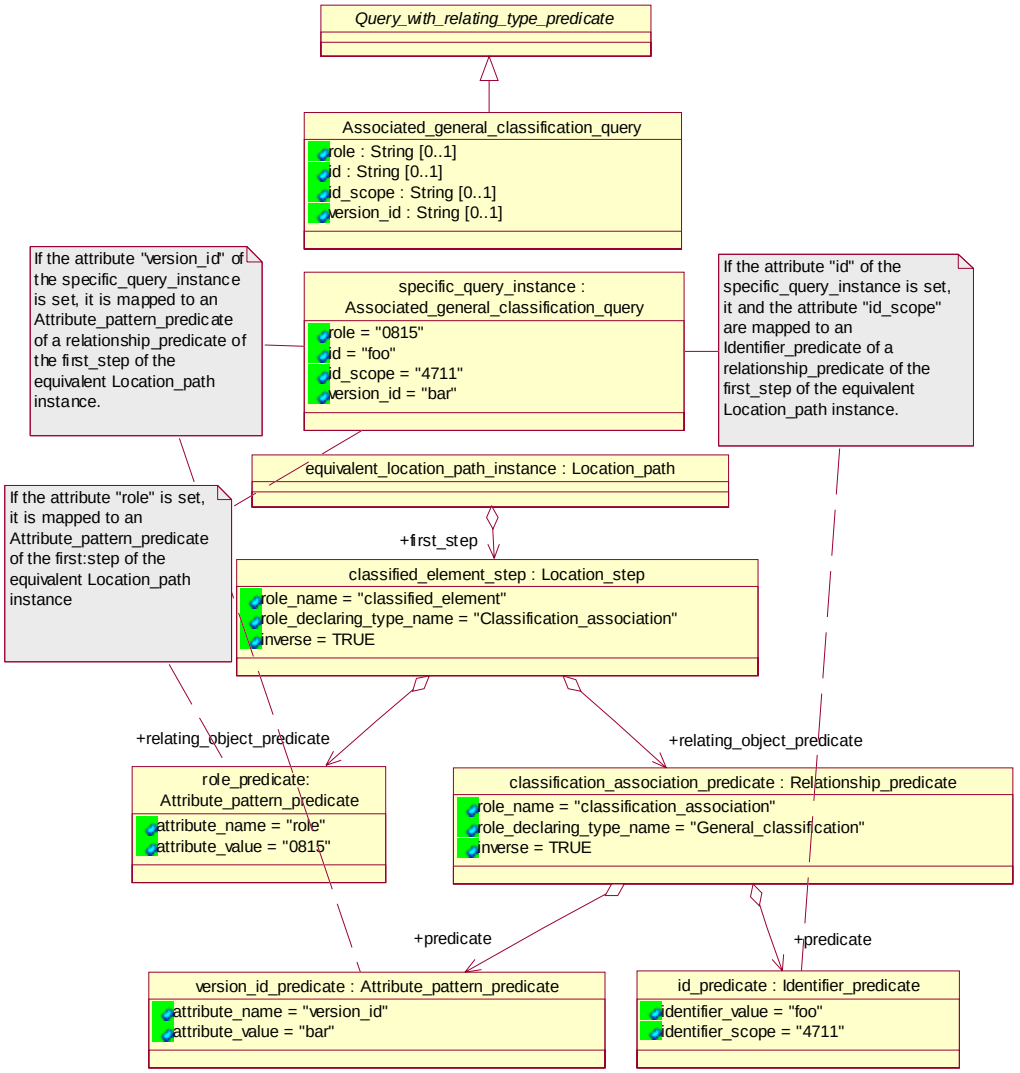


**Figure 9.36 - Definition, sample instance and equivalent Location_path instance of the Associated_general_classification_query**

## 9.7.21 Associated_item_property_query

The Associated_item_property_query traverses from Item_property_select objects via Property_value_association objects and Property_value_representation objects to the associated Property_value objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- value_name : String [0..1]
- relating_type_name : String [0..*]

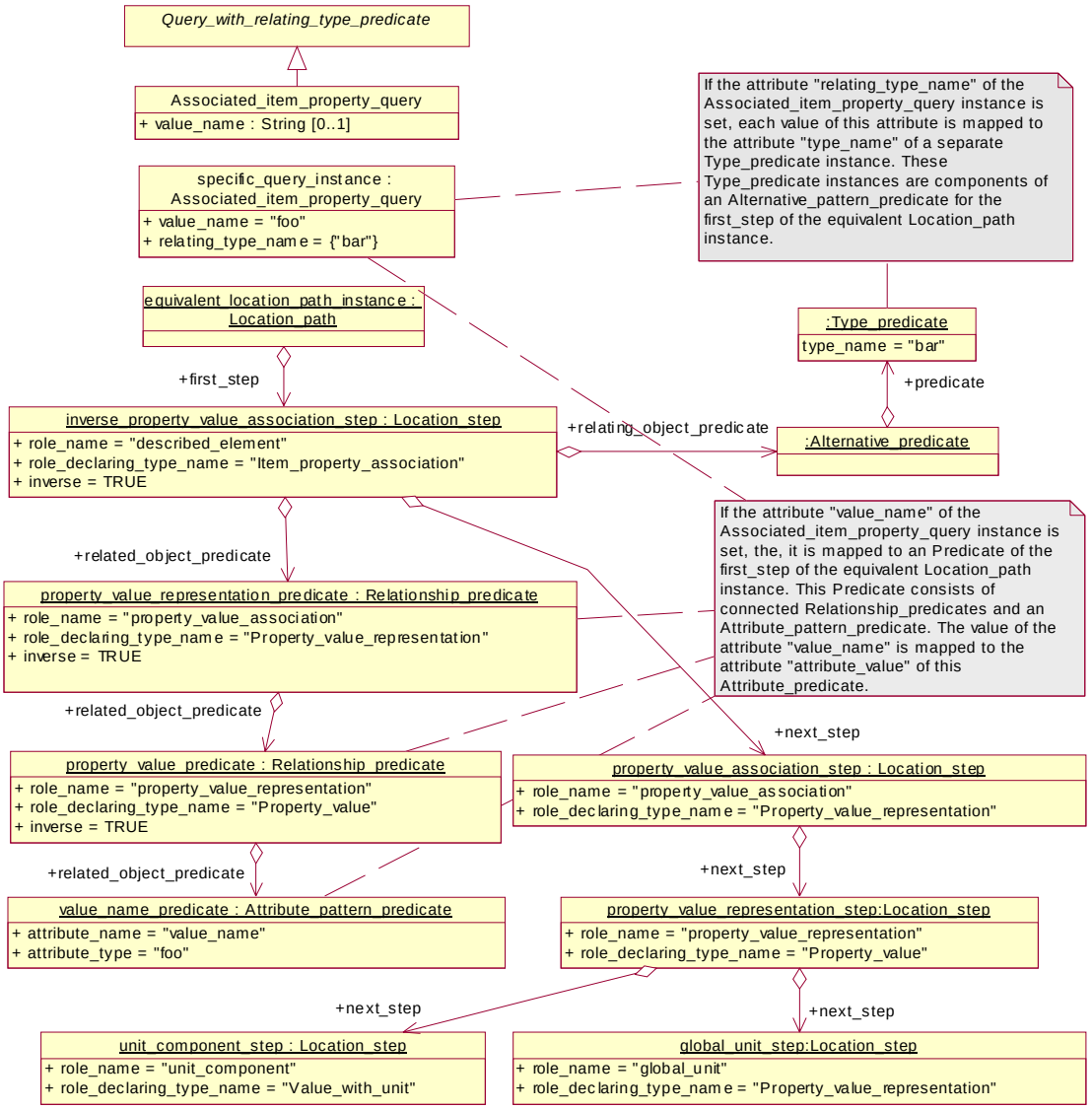


**Figure 9.37 - Definition, sample instance and equivalent Location_path instance of the Associated_item_property_query**

### 9.7.22 Associated_person_organization_query

The Associated_person_organization_query traverses from Date_time_person_organization_element_select objects via Person_organization_assignment objects to Person_organization_select objects.

#### *Base Class*

- Query_with_relating_type_predicate

#### *Parameters*

- role : String [0..1]
- relating_type_names : String [0..*]

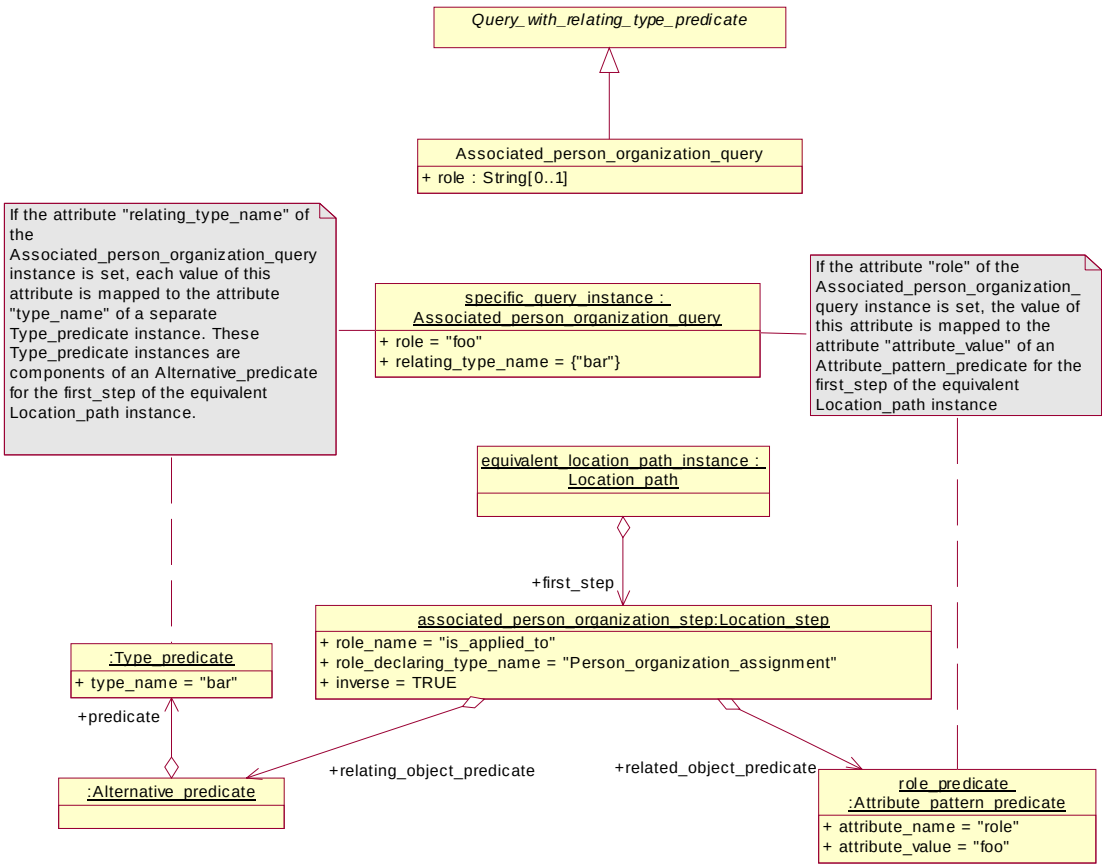


**Figure 9.38 - Definition, sample instance and equivalent Location_path instance of the Associated_person_organization_query**

### 9.7.23 Associated_process_property_query

The Associated_process_property_query traverses from Process_property_select objects via Property_value_association objects and Property_value_representation objects to the associated Property_value objects.

#### *Base Class*

- Query_with_relating_type_predicate

#### *Parameters*

- value_name : String [0..1]
- relating_type_name : String [0..*]

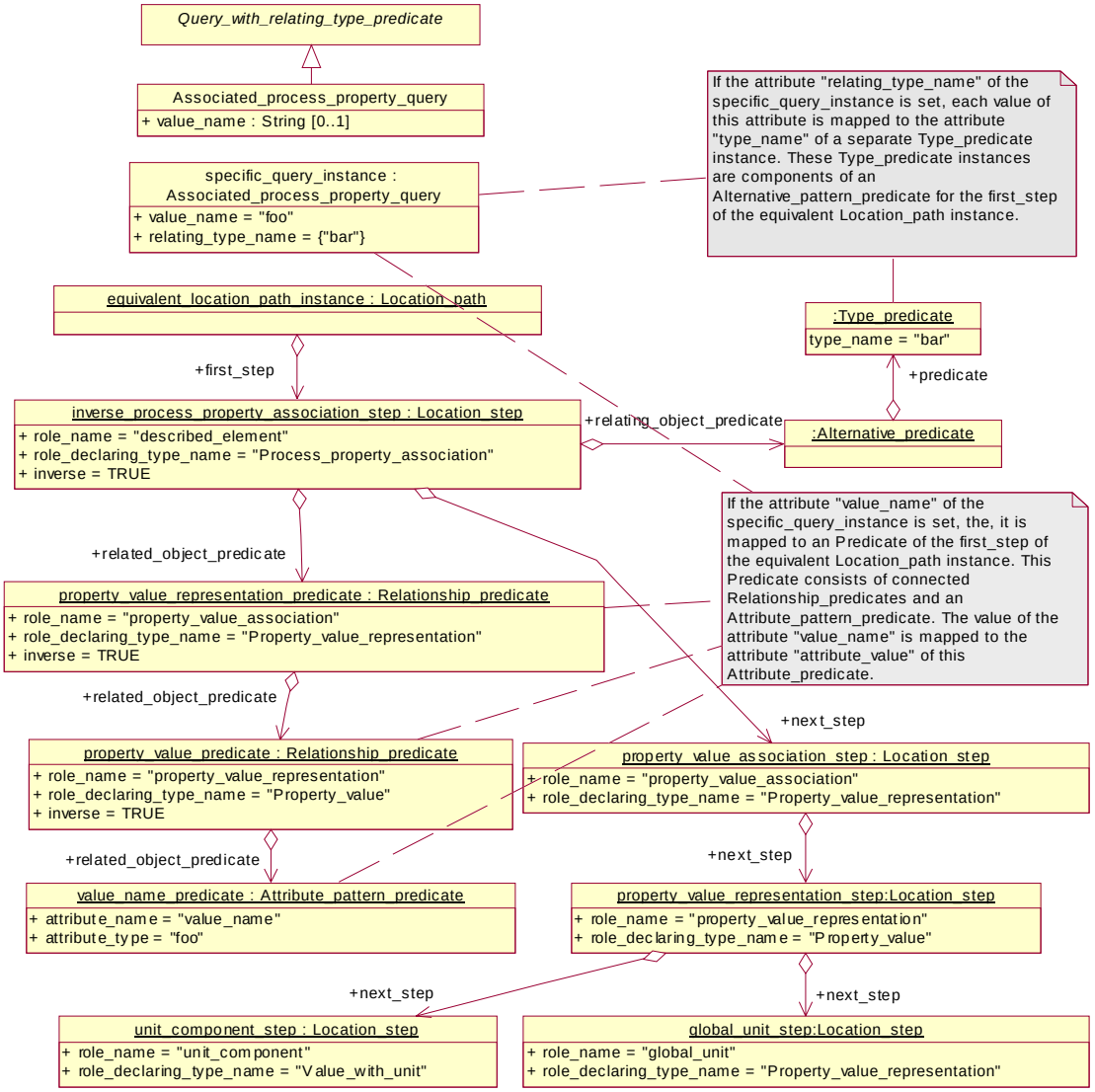


**Figure 9.39 - Definition, sample instance and equivalent Location_path instance of the Associated_process_property_query**

## 9.7.24 Associated_project_ query

The Associated_project_query traverses from Project_information_select objects via Project_assignment objects to Project objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- role : String [0..1]
- relating_type_name : String [0..*]

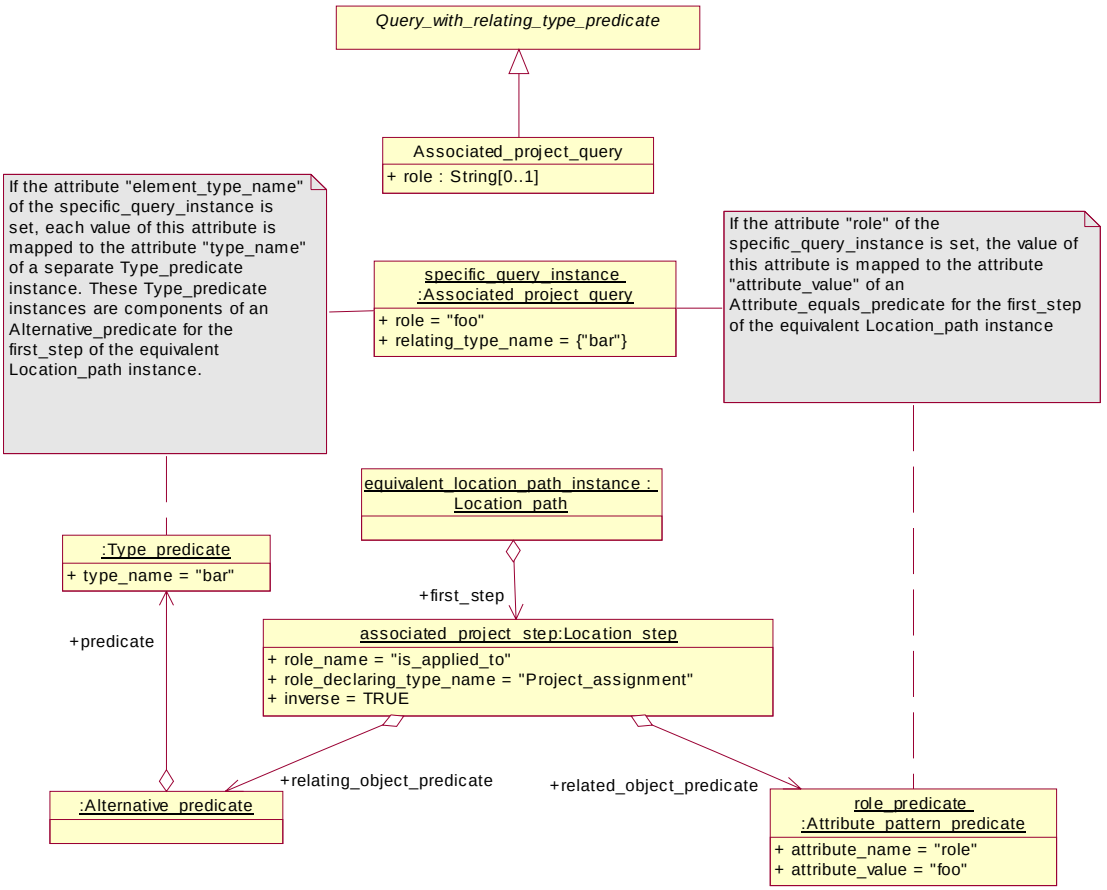


**Figure 9.40 - Definition, sample instance and equivalent Location_path instance of the Associated_project_query**

## 9.7.25 Associated_property_query

The Associated_property_query traverses from Item_property_select and Process_property_select objects via Property_value_association objects and Property_value_representation objects to the associated Property_value objects.

The Associated_property_query is defined as a Batch_query of an Associated_item_property_query and an Associated_process_property_query.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- value_name : String [0..1]
- relating_type_name : String [0..*]

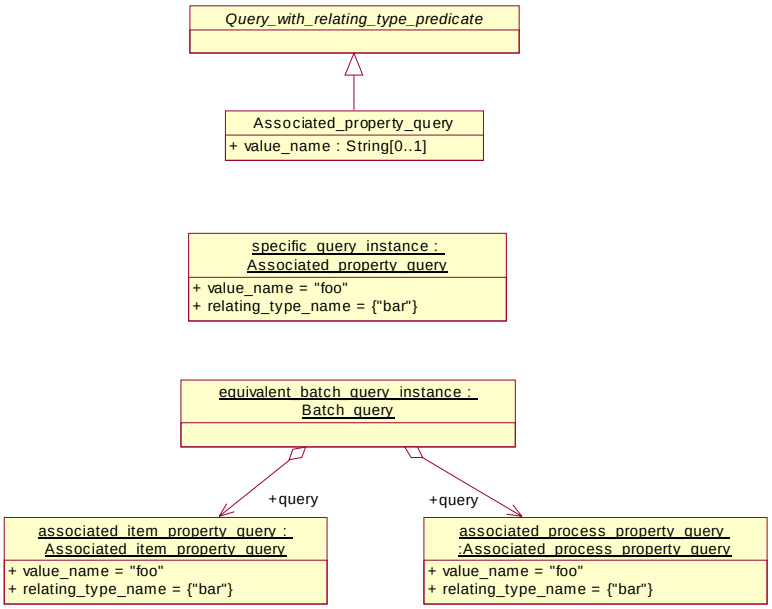


**Figure 9.41 - Definition, sample instance and equivalent Location_path instance of the Associated_property_query**

## 9.7.26 Class_structure_query

The Class_structure_query traverses from Product_class objects via Class_structure_relationship objects to Product_function_component_select objects.

### *Base Class*

- Specific_query

### *Parameters*

- relation_type : String [0..1]

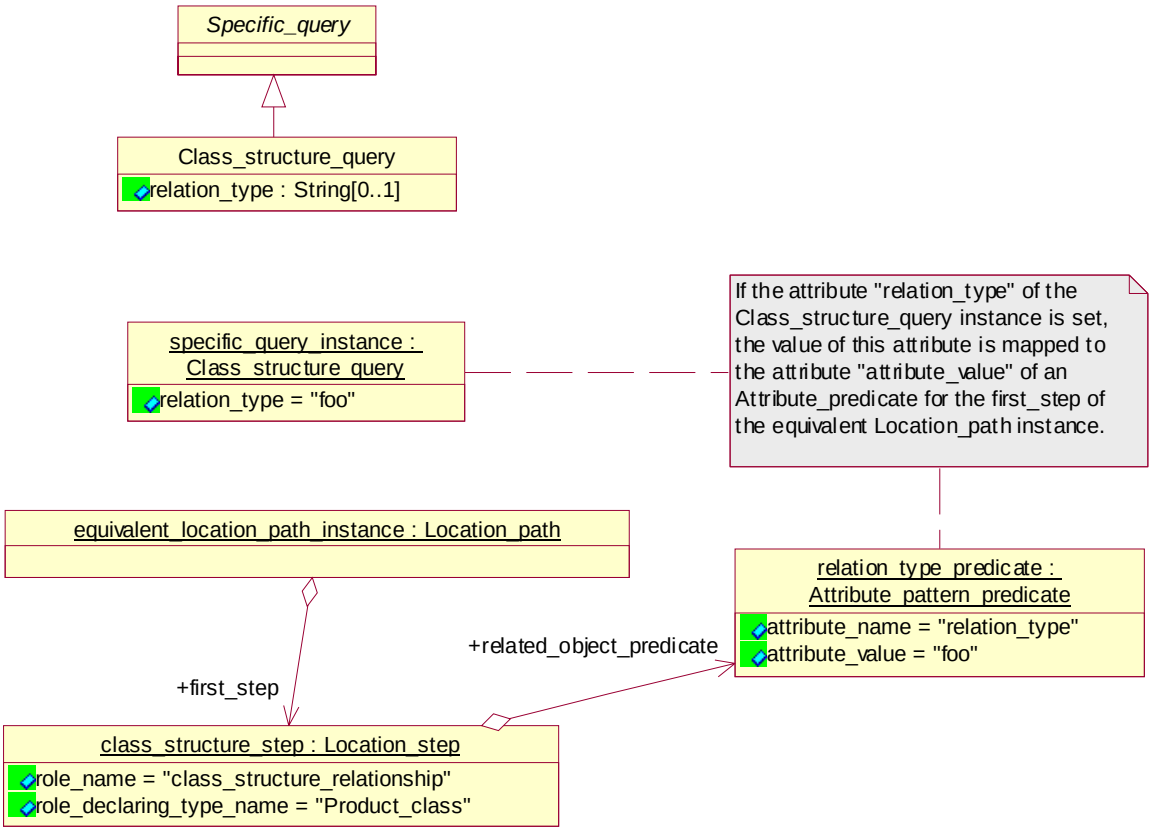


**Figure 9.42 - Definition, sample instance and equivalent Location_path instance of the Class_structure_query**

### 9.7.27 Complex_product_query

The Complex_product_query selects Complex_product objects by its id and version_id atrtributes.

#### *Base Class*

- Specific_query

#### *Parameters*

- id : String [0..1]
- id_scope : String [0..1]
- version_id : String [0..1]

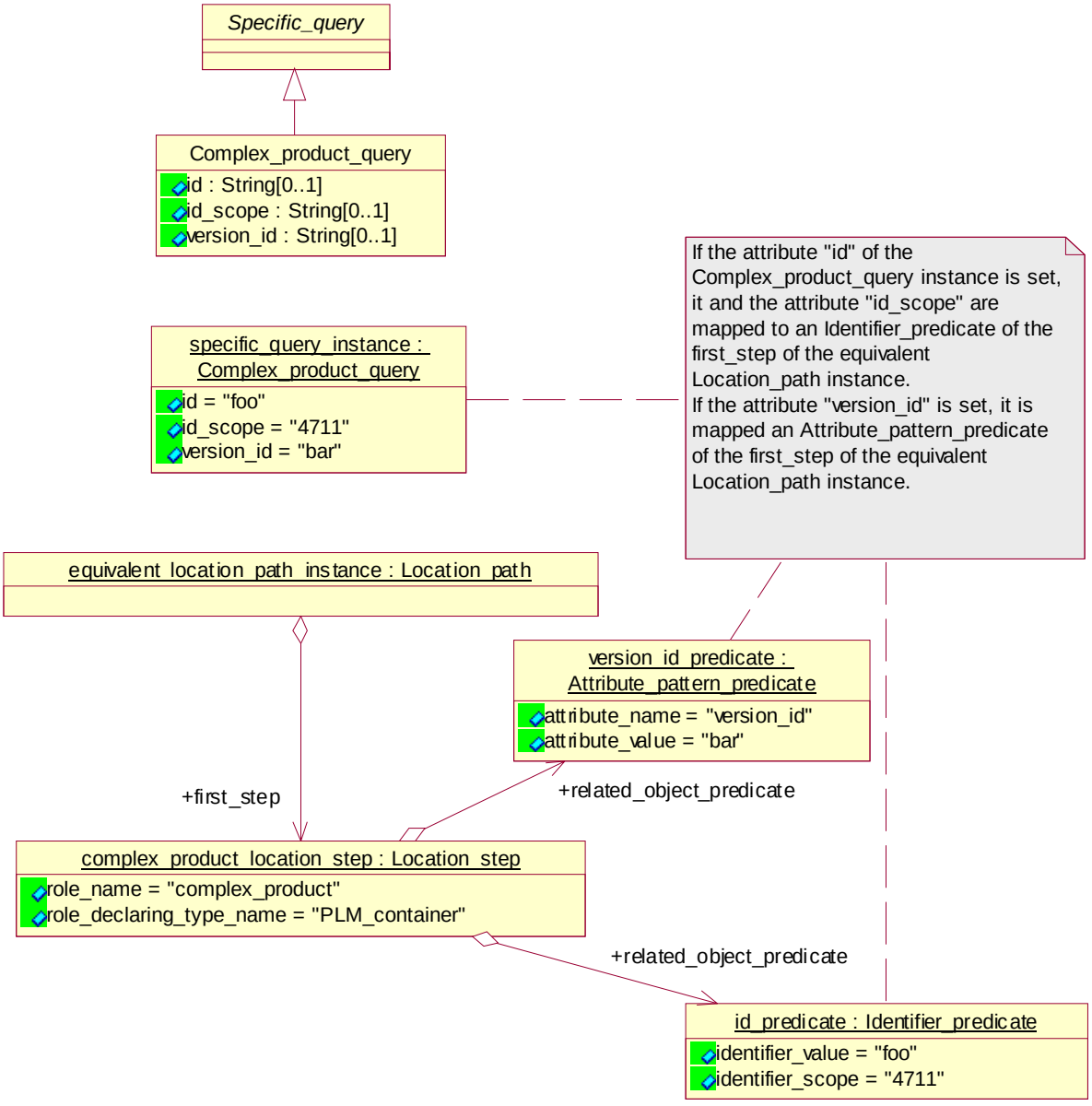


**Figure 9.43 - Definition, sample instance and equivalent Location_path instance of the Complex_product_query**

### 9.7.28 Configuration_query

The Configuration_query traverses from Configured_item_select objects via Configuration objects to Configured_specification_select objects.

### *Base Class*

- Query_with_relating_type_predicate

### *Parameters*

- configuration_type : String [0..1]
- inheritance_type : String [0..1]
- relating_type_name : String [0..*]

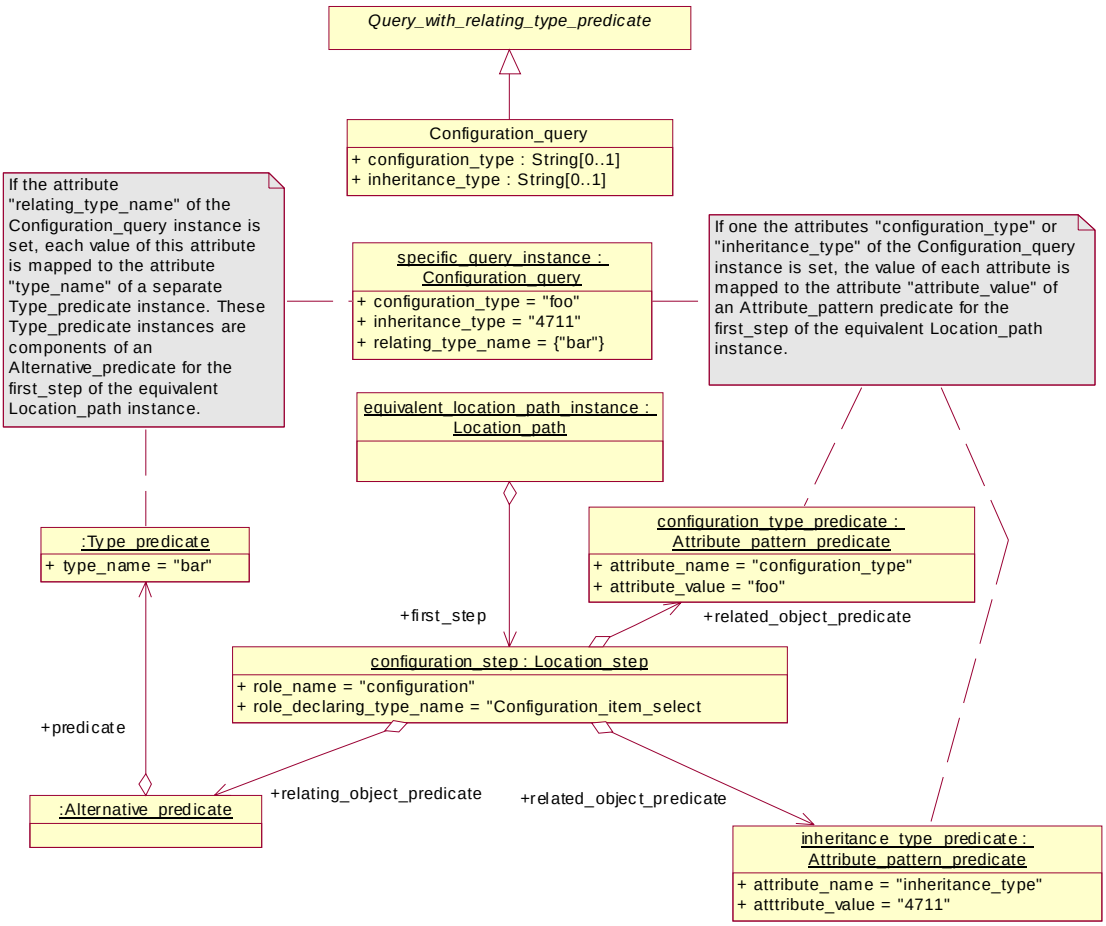


**Figure 9.44 - Definition, sample instance and equivalent Location_path instance of the Configuration_query**

## 9.7.29 Date_organization_assignment_query

The Date_organization_assignment_query traverses from Date_and_person_organization objects via Date_and_person_assignment objects to Date_time_person_organization_element_select objects.

### *Parameters*

- role : String [0..1]
- related_type_name : String [0..*]

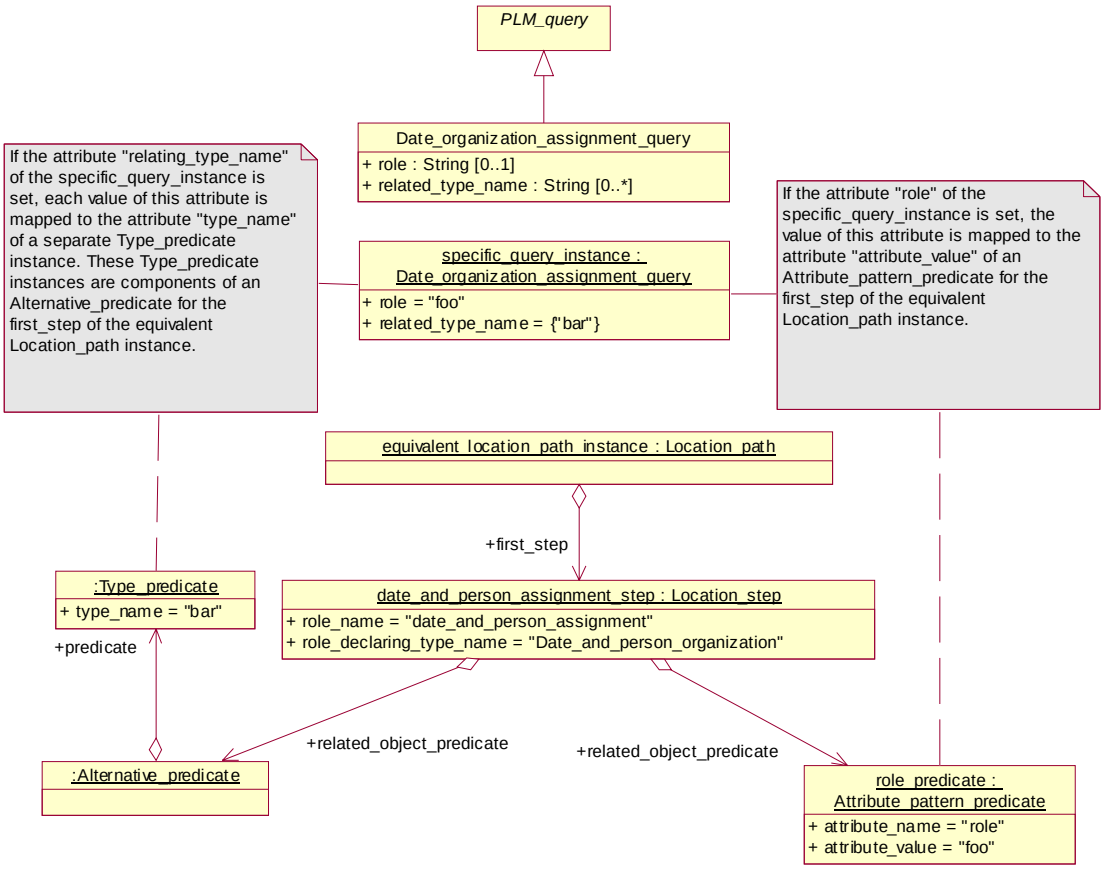


**Figure 9.45 - Definition, sample instance and equivalent Location_path instance of the Date_organization_assignment_query**

## 9.7.30 Design_discipline_item_definition_query

The Design_discipline_item_definition_query traverses from Item_version objects to Design_discipline_item_definition objects.

### *Parameters*

- id : String [0..1]
- application_domain : String [0..1]

  traverse only Design_discipline_item_definition objects which relates via their

initial_context association to an Application_context object with an application_domain attribute of the given value

- life_cycle_stage : String [0..1] traverse only Design_discipline_item_definition objects which relates via their initial_context association to an Application_context object with an life_cycle_stage attribute of the given valu

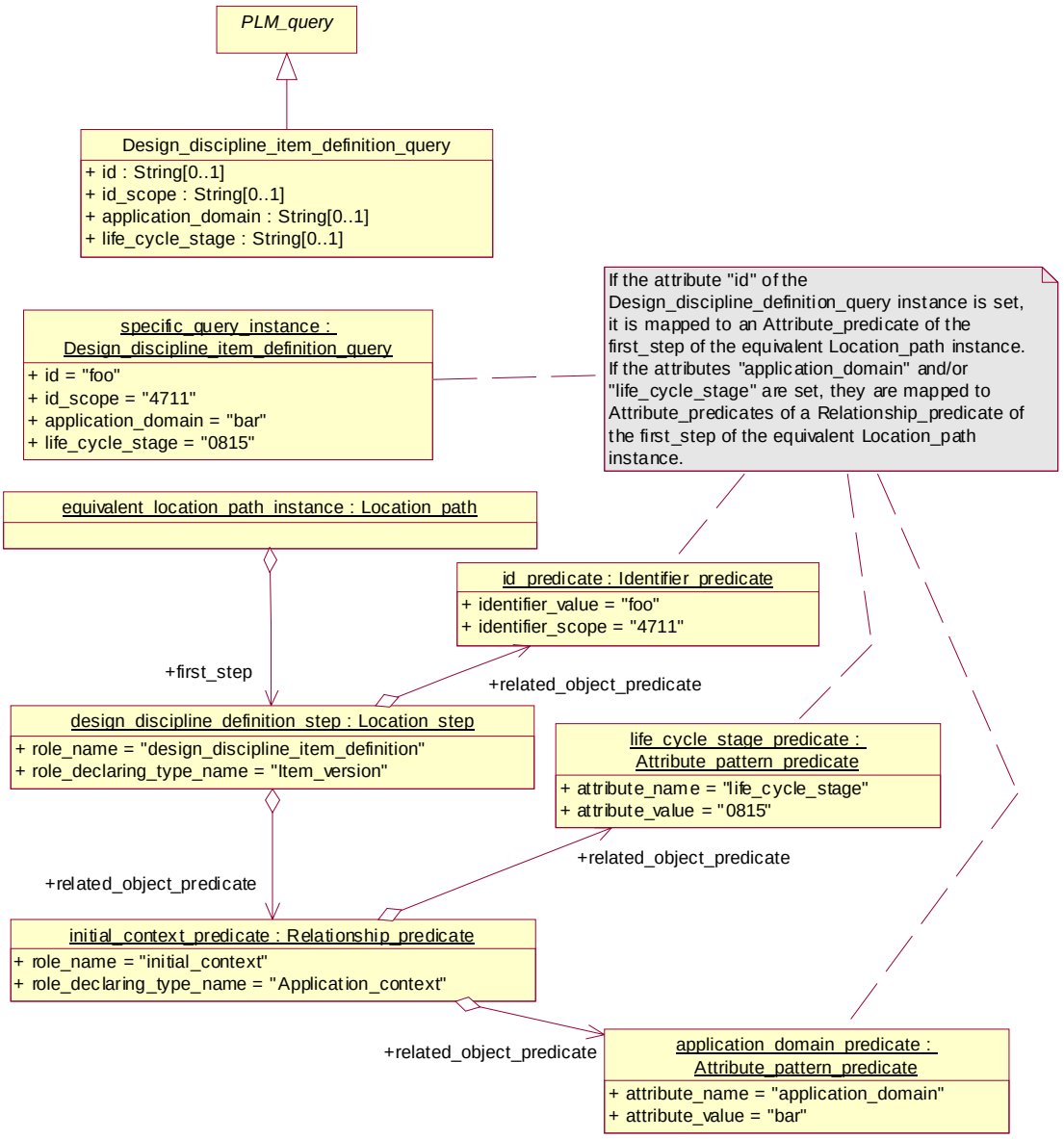


**Figure 9.46 - Definition, sample instance and equivalent Location_path instance of the Design_discipline_item_definition_query**

### 9.7.31 Document_classification_query

The Document_classification_query traverses from Document objects to Specific_document_classification objects.

#### *Parameters*

- classification_name : String [0..1]

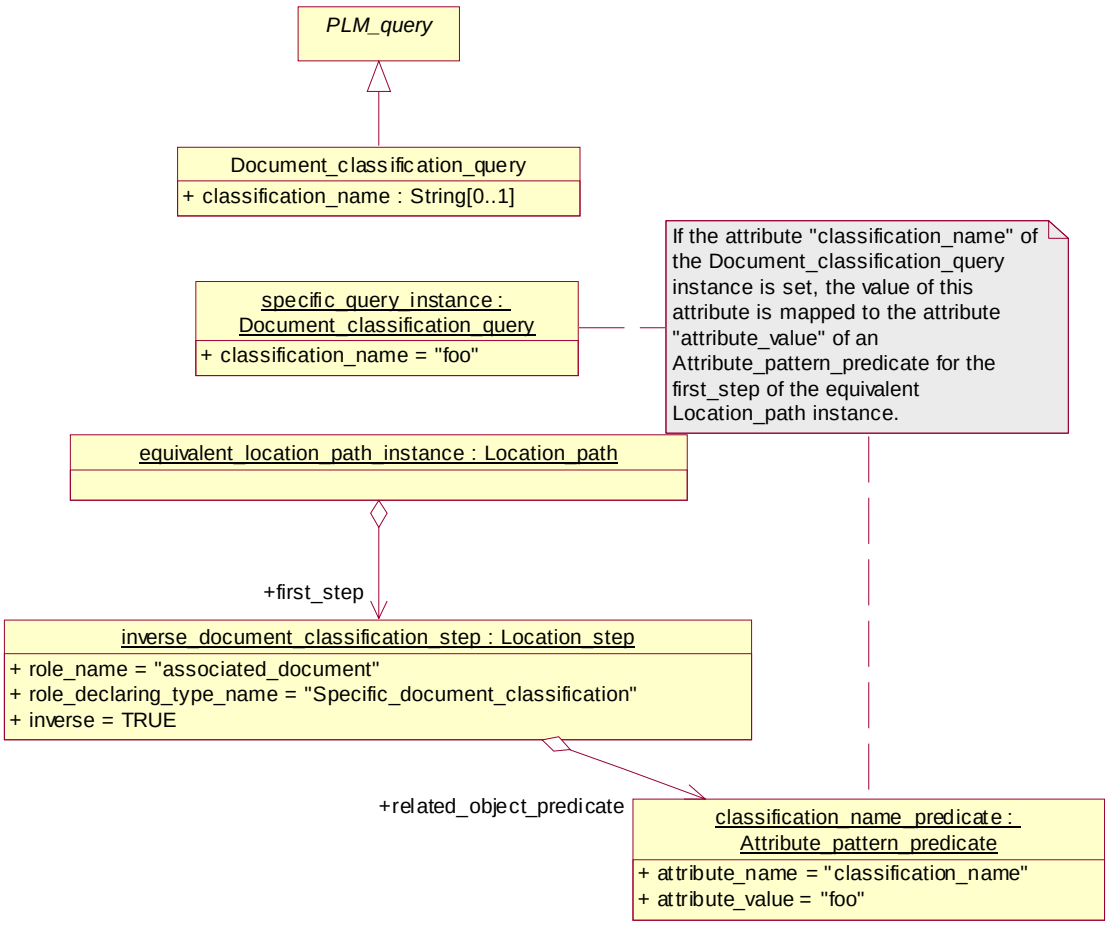


**Figure 9.47 - Definition, sample instance and equivalent Location_path instance of the Document_classification_query**

### 9.7.32 Document_classification_hierarchy_query

The Document_classification_hierarchy_query traverses from Specific_document_classification objects to Specific_document_classification objects via Specific_ocument_classification_hierarchy objects

#### *Base Class*

- Specific_query

### *Parameters*

- inverse: boolean [0..1]

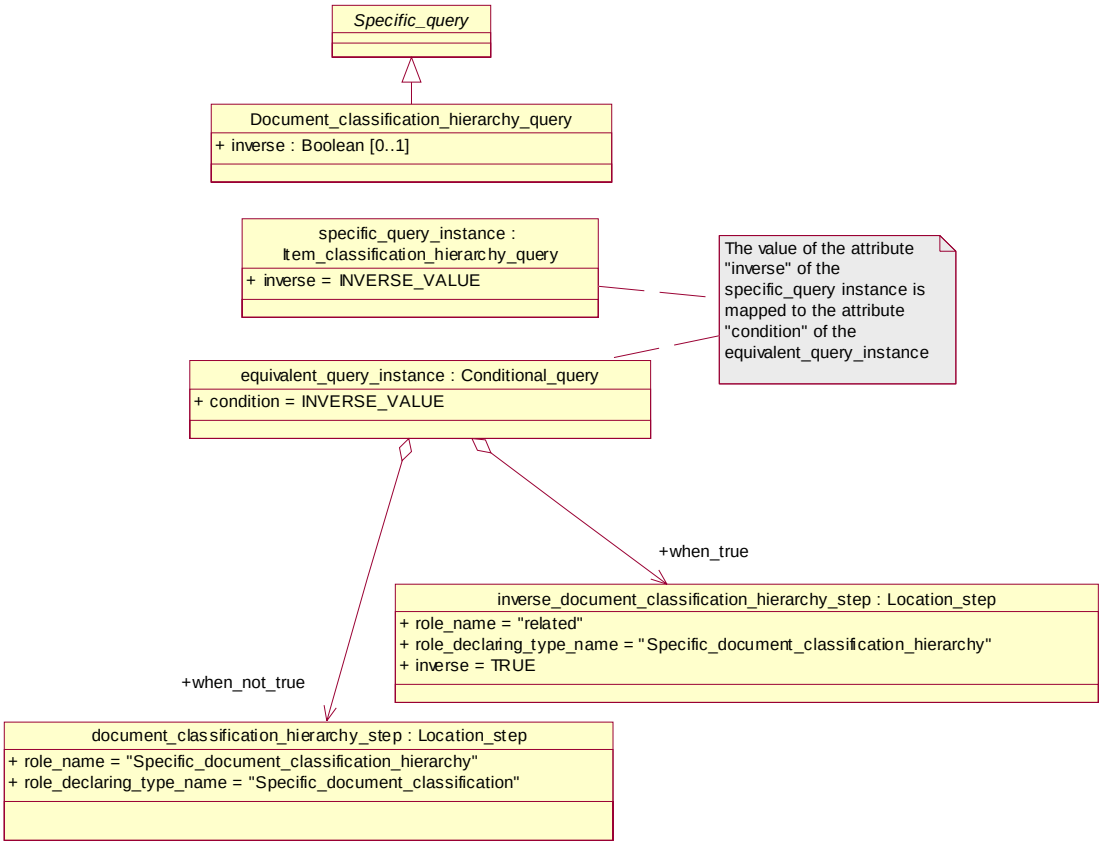


**Figure 9.48 - Definition, sample instance and equivalent Location_path instance of the Document_classification_hierarchy_query**

## 9.7.33 Document_property_query

The Document_property_query traverses the document properties from Docu-ment_representation objects.

These properties are Document_creation_property, Document_size_property, Document_format_property, Document_content_property, and Document_location_property.

### *Base Class*

- Specific_query

### *Parameters*

- none

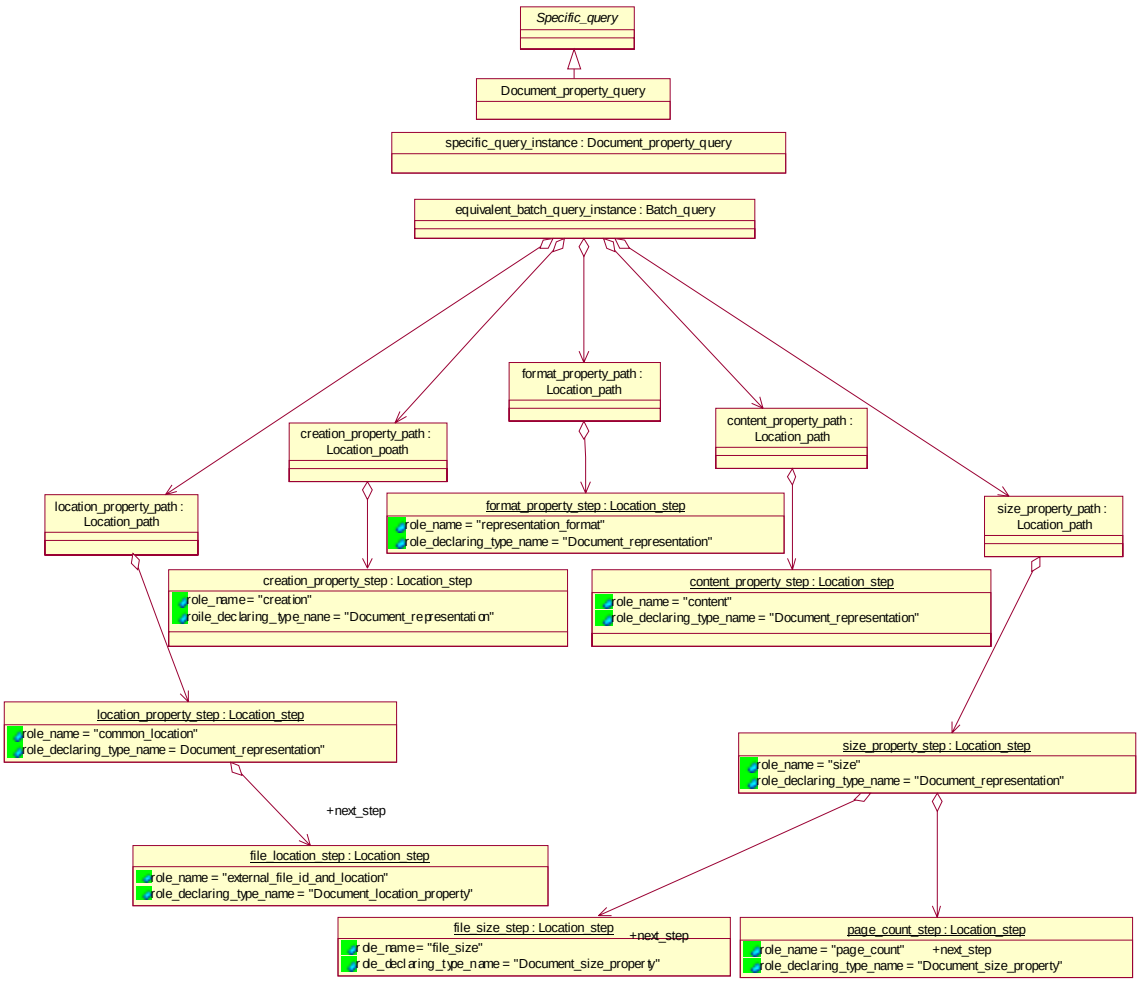


**Figure 9.49 - Definition, sample instance and equivalent Location_path instance of the Document_property_query**

## 9.7.34 Document_query

The Document_query selects Document objects.

### *Parameters*

- document_id : String [0..1]
- document_id_scope : String [0..1]
- name : String [0..1]
- name_language : Language [0..1]
- version_id : String [0..1]
- classification_name : String [0..1]

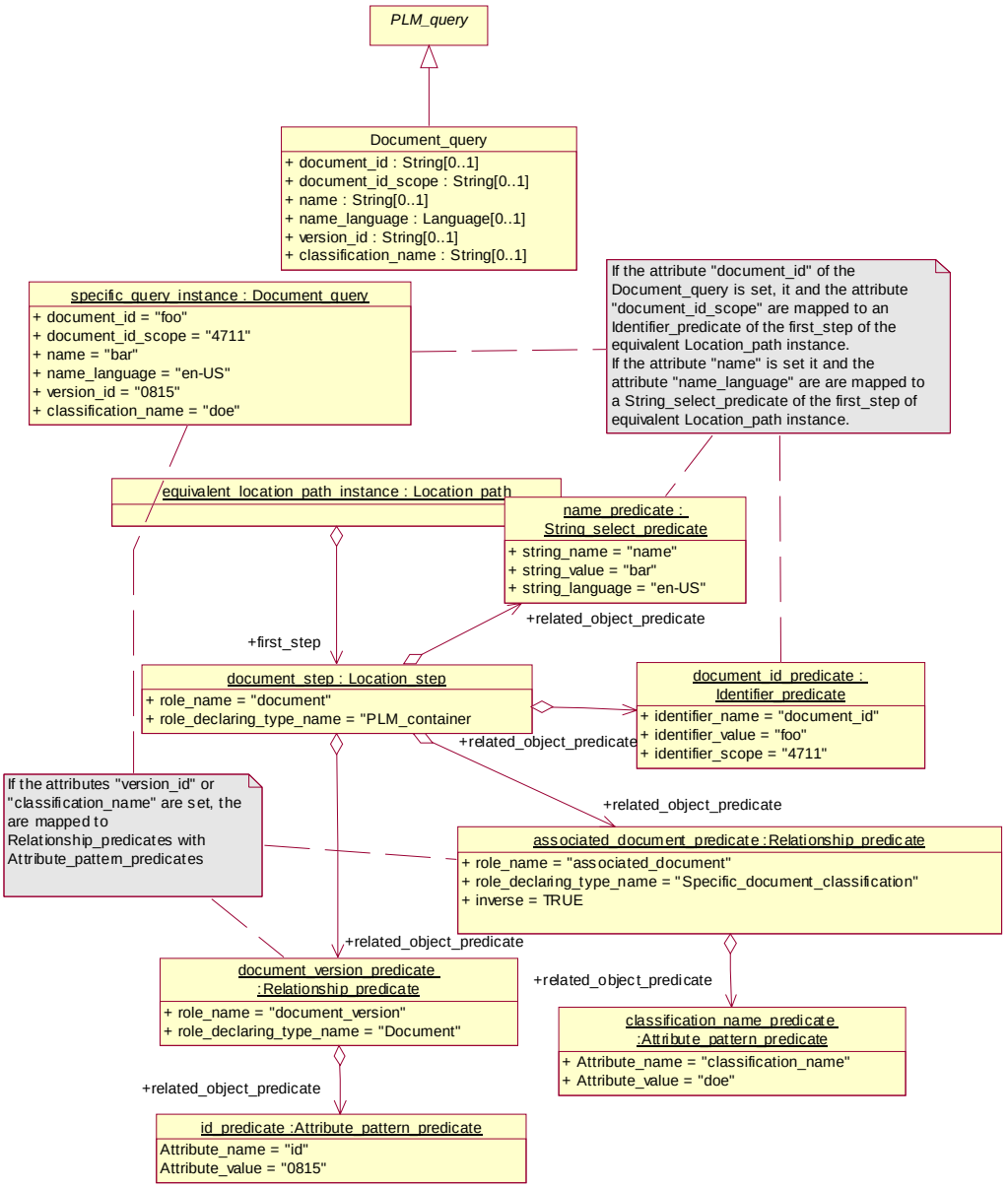


**Figure 9.50 - Definition, sample instance and equivalent Location_path instance of the Document_query**

## 9.7.35 Document_representation_query

The Document_representation_query traverses Document_representation objects from Document_version objects.

***Parameters***

- id : String [0..1]
- id_scope : String [0..1]

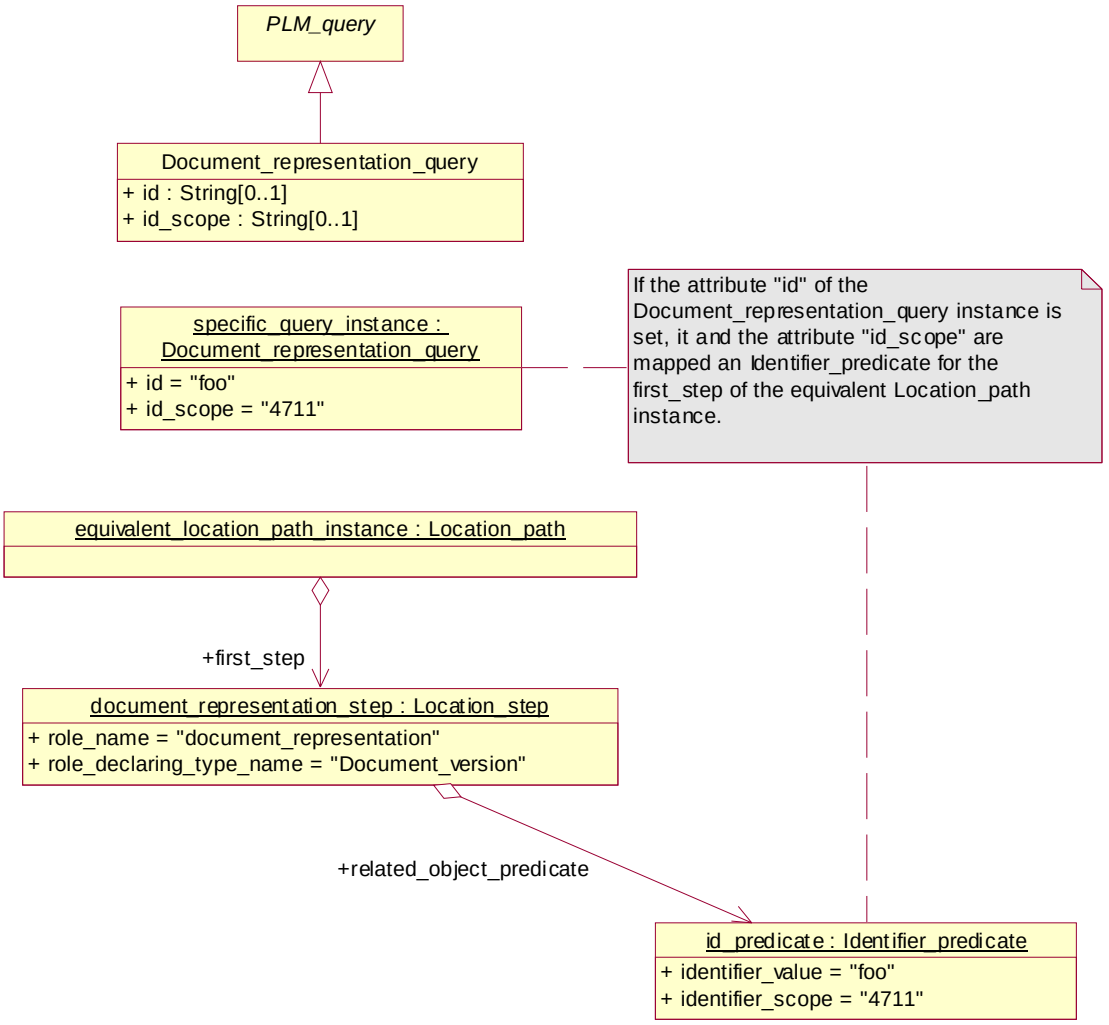


**Figure 9.51 - Definition, sample instance and equivalent Location_path instance of the Document_representation_query**

## 9.7.36 Document_structure_query

The Document_structure_query traverses from Document_representation objects via Document_structure objects to Document_representation objects.

### *Base Class*

- Releationship_query

### *Parameters*

- relation_type : String [0..1]
- inverse : Boolean [0..1]

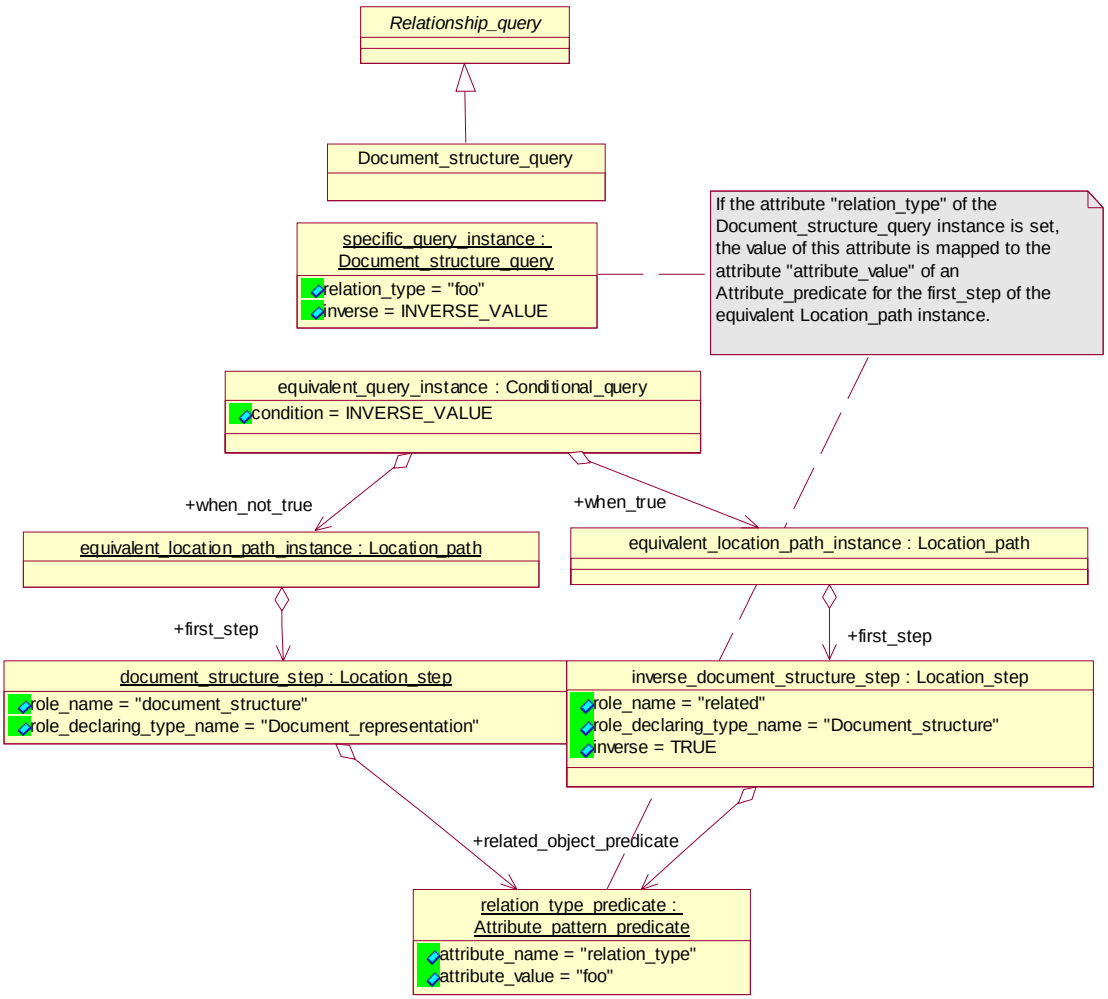


**Figure 9.52 - Definition, sample instance and equivalent Location_path instance of the Document_structure_query**

## 9.7.37 Document_version_query

The Document_version_query traverses Document_version objects of Document objects.

### *Parameters*

- id : String [0..1]
- id_scope : String [0..1]

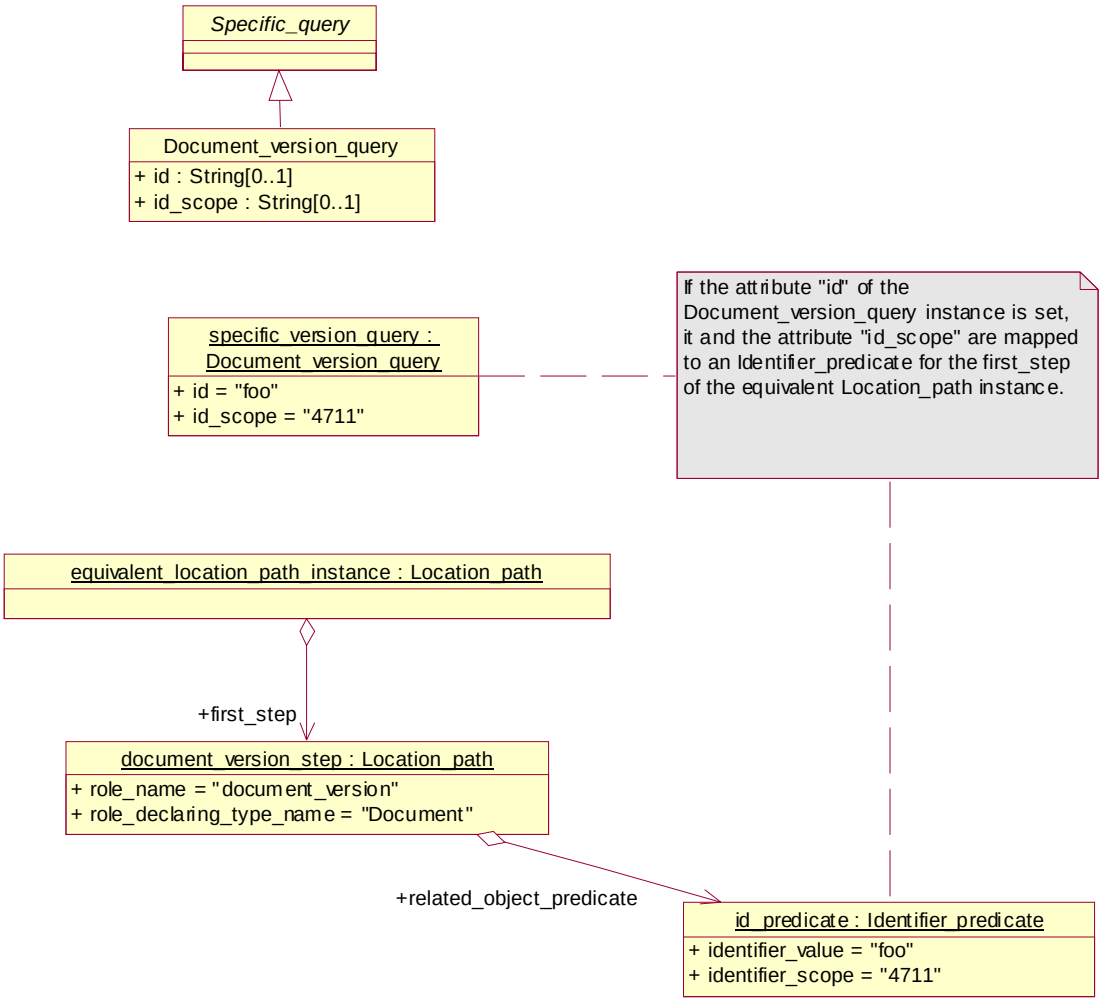


**Figure 9.53 - Definition, sample instance and equivalent Location_path instance of the Document_version_query**

## 9.7.38 Effectivity_assignment_query

The Effectivity_assignment_query traverses from Effectivity objects to Effectivity_element_select objects via Effectivity_assignment objects.

### *Base Class*

- Specific_query

### *Parameters*

- Role: string [0..1]
- Effectivity_indication: boolean [0..1]

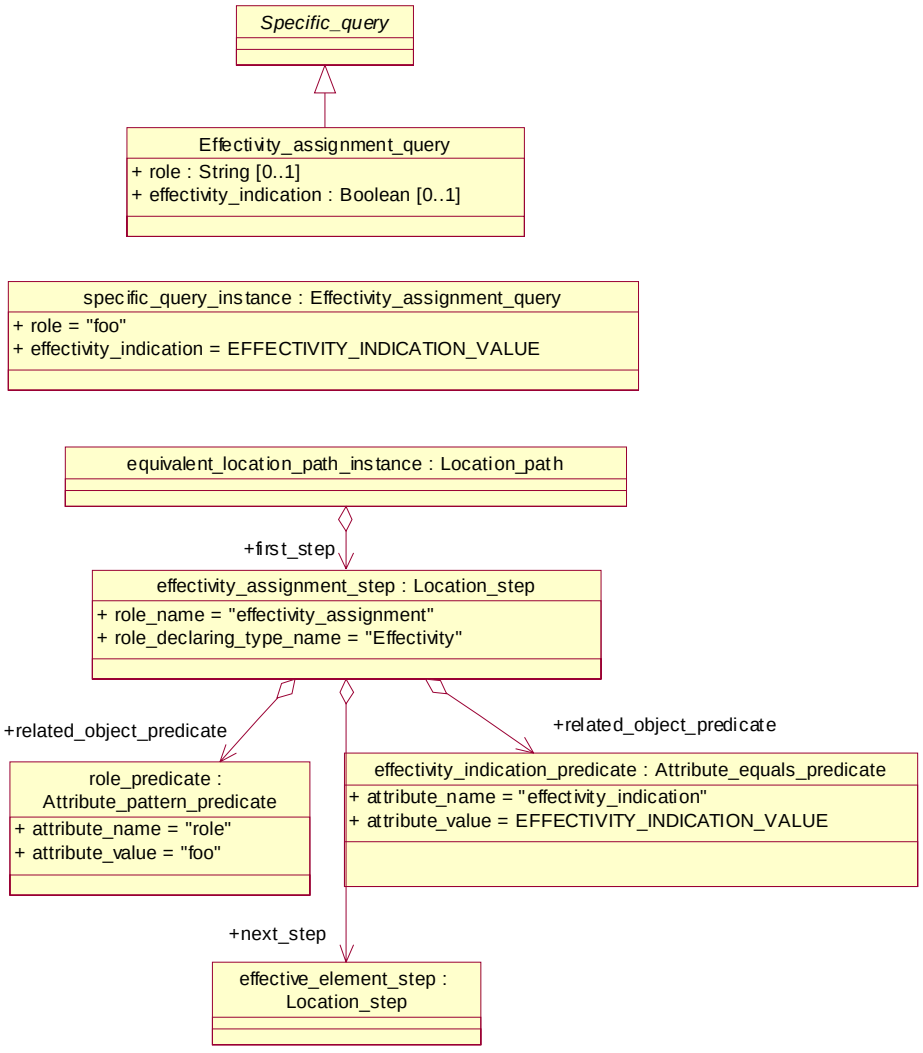


**Figure 9.54 - Definition, sample instance and equivalent Location_path instance of the Effectivity_assignment_query**

## 9.7.39 Effectivity_query

The Effectivity_query selects Effectivity objects.

### *Parameters*

- id : String [0..1]

- Id_scope : String [0..1]
- version_id : String [0..1]
- effectivity_context : String [0..1]

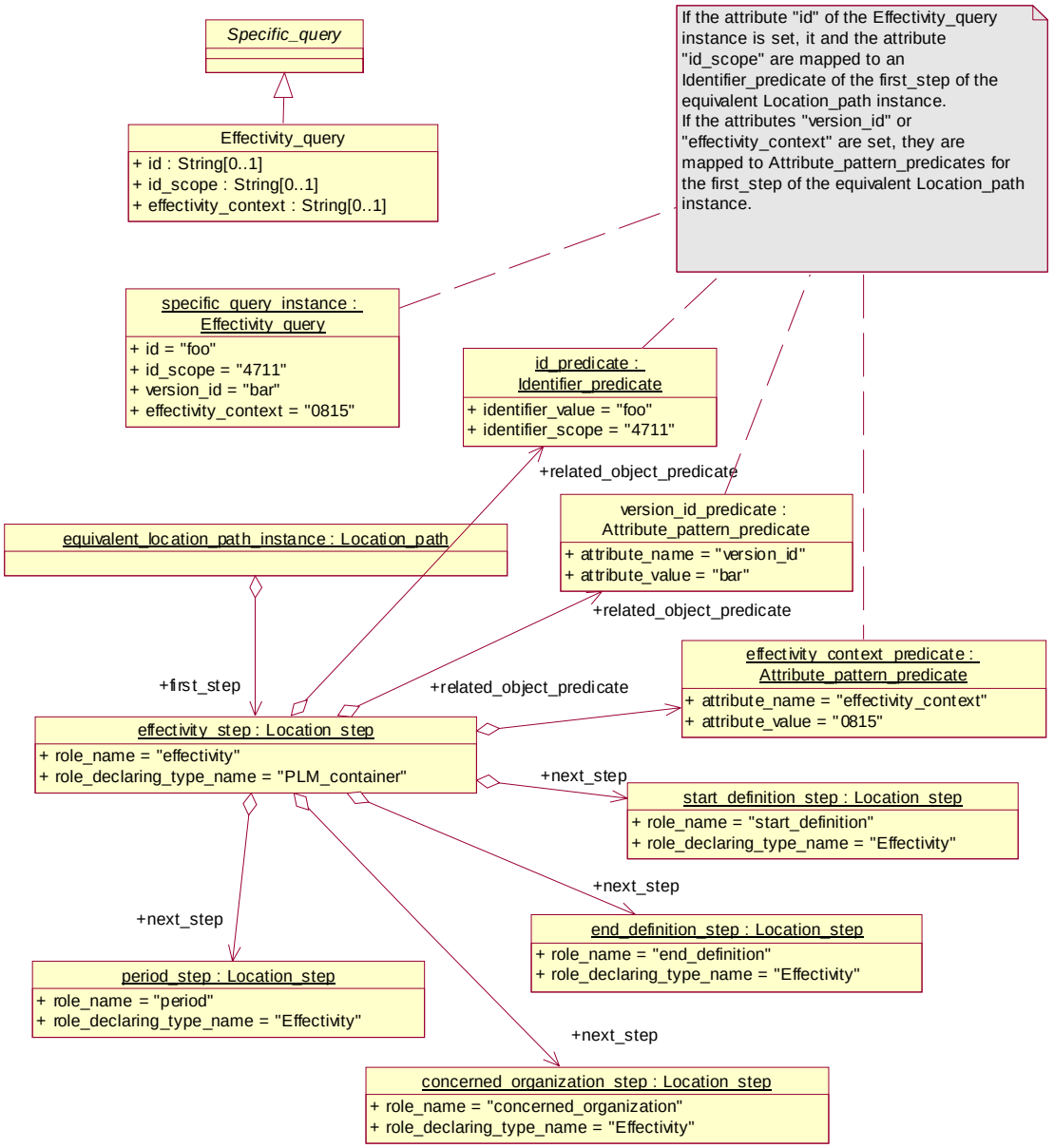


**Figure 9.55 - Definition, sample instance and equivalent Location_path instance of the Effectivity_query**

### 9.7.40 Event_reference_query

The Event_reference_query selects Event_reference objects and traverses from the Event_reference_objects to their offset and event_context references.

#### *Base Class*

- Specific_query

#### *Parameters*

- type: string [0..1]
- add_event_context: boolean [0..1]
- add_offset: boolean [0..1]

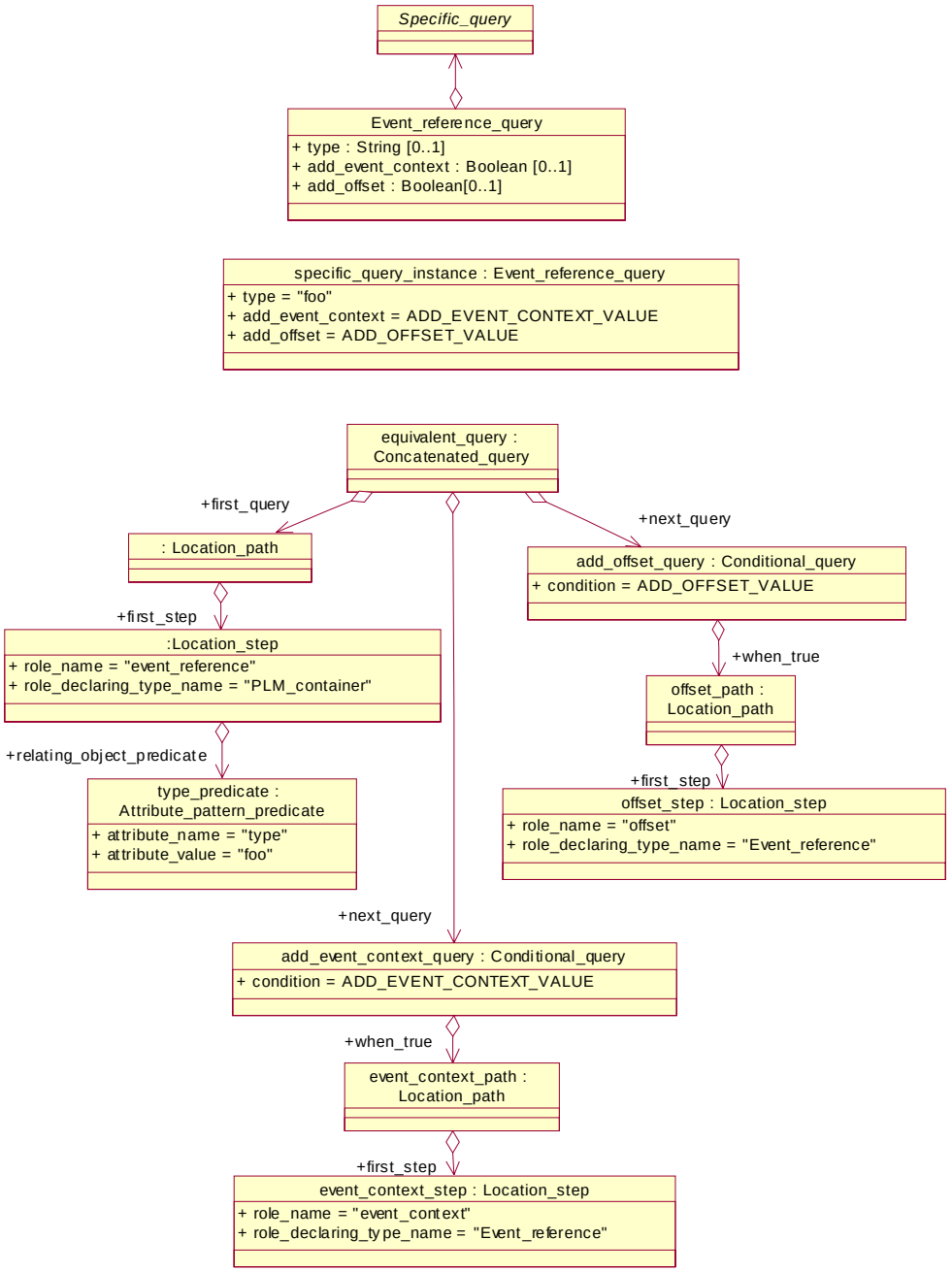


**Figure 9.56 - Definition, sample instance and equivalent Location_path instance of the Event_reference_query**

### 9.7.41 External_model_query

The External_model_query traverses from Digital_file objects to External_model objects.

#### *Base Class*

- Specific_query

#### *Parameters*

- Model_id: string [0..1]

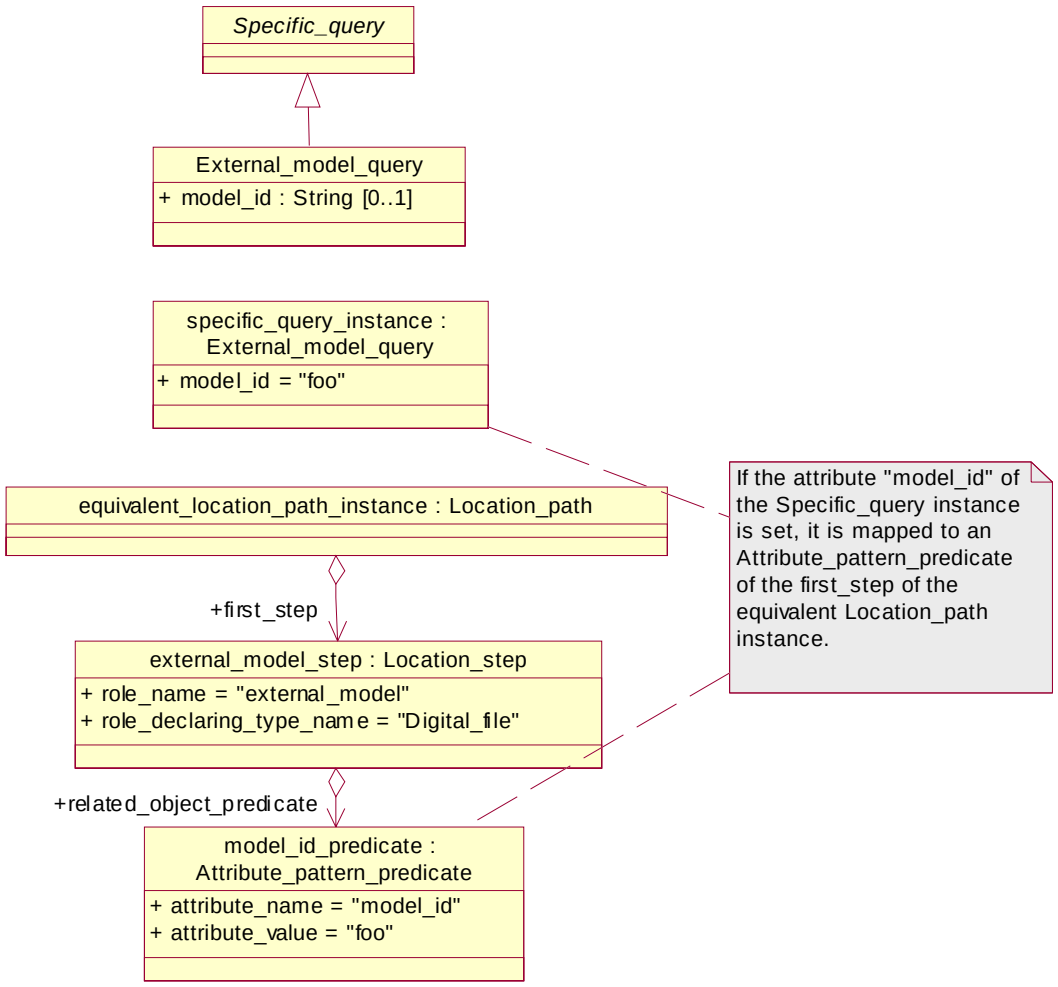


**Figure 9.57 - Definition, sample instance and equivalent Location_path instance of the External_model_query**

### 9.7.42 File_property_query

The File_property_query traverses from Document_file objects their file property objects.

#### *Base Class*

- Specific_query

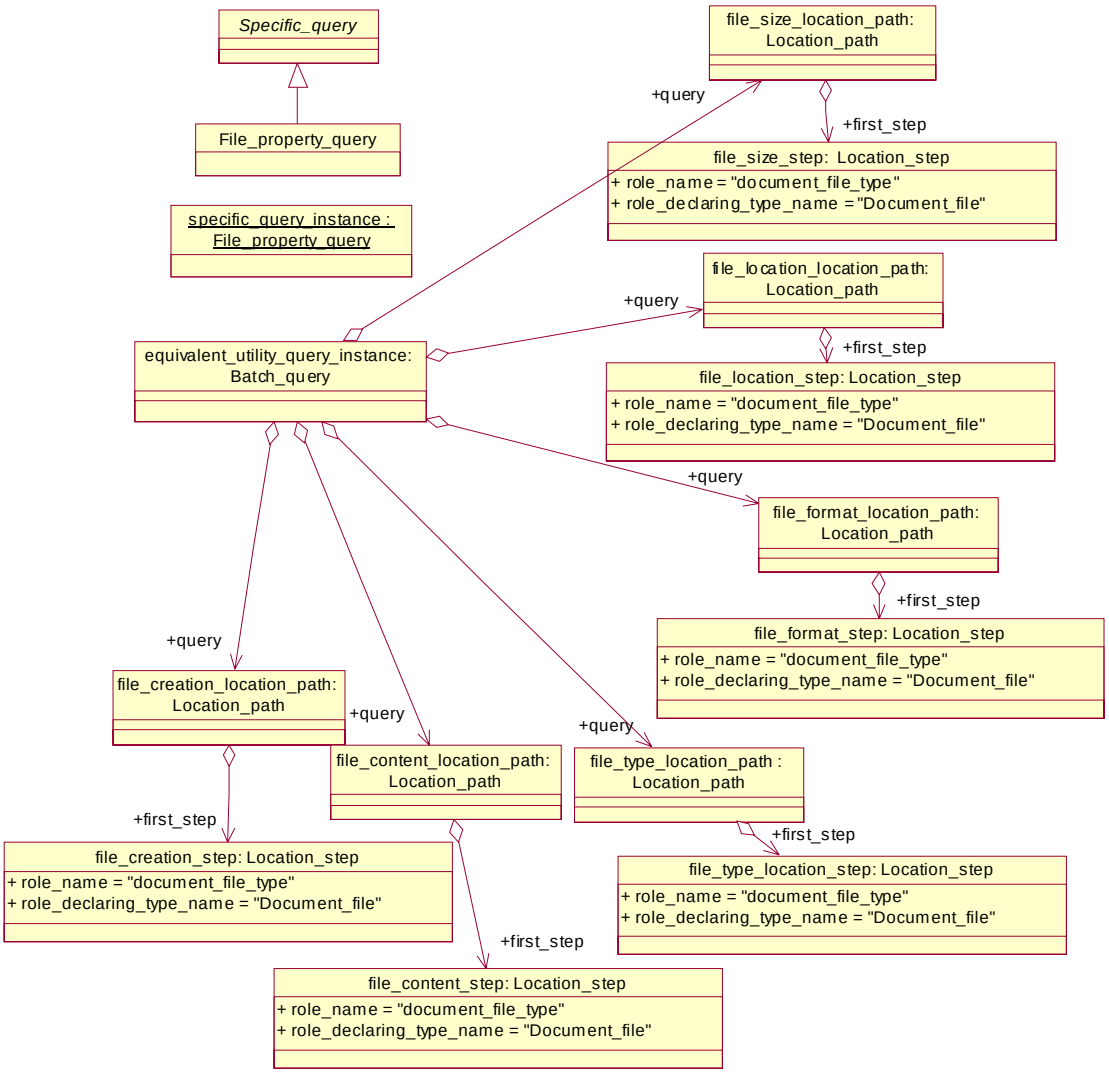


**Figure 9.58 - Definition, sample instance and equivalent Location_path instance of the File_property_query**

### 9.7.43 Geometric_model_relationship_query

The query traverses from Geometric_or_external_model_select objects to Geometric_or_external_model_select_ objects via Geometric_model_relationship objects.

#### *Base Class*

- Relationship_query

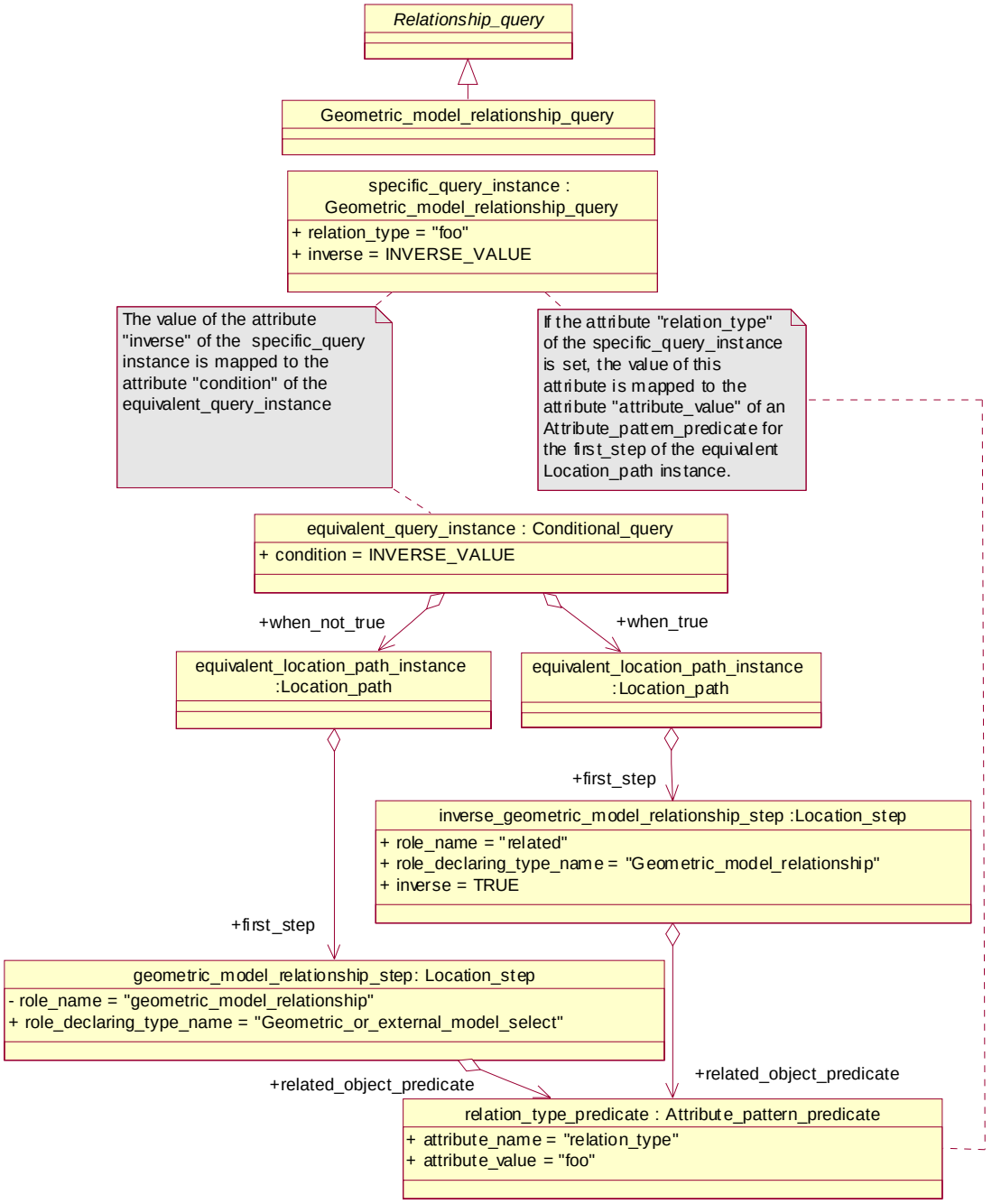


**Figure 9.59 - Definition, sample instance and equivalent Location_path instance of the Geometric_model_relationship_query**

### 9.7.44 Item_classification_query

The Item_classification_query traverses the Specific_item_classification objects from Item objects.

#### *Parameters*

- classification_name : String [0..1

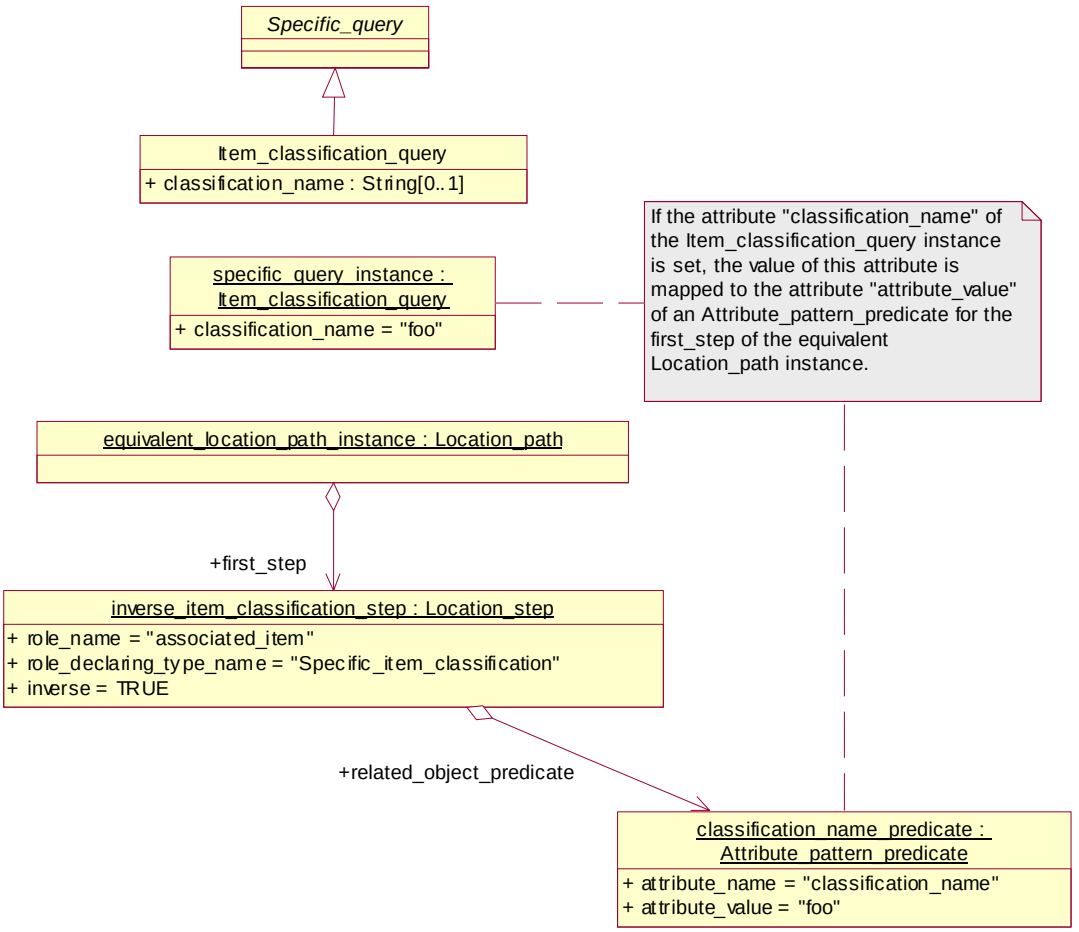


**Figure 9.60 - Definition, sample instance and equivalent Location_path instance of the Item_classification_query**

### 9.7.45 Item_classification_hierarchy_query

The Item_classification_hierarchy_query traverses from Specific_item_classification objects to Specific_item_classification objects via Specific_item_classification_hierarchy objects.

#### *Base Class*

- Specific_query

### *Parameters*

- inverse: boolean [0..1]

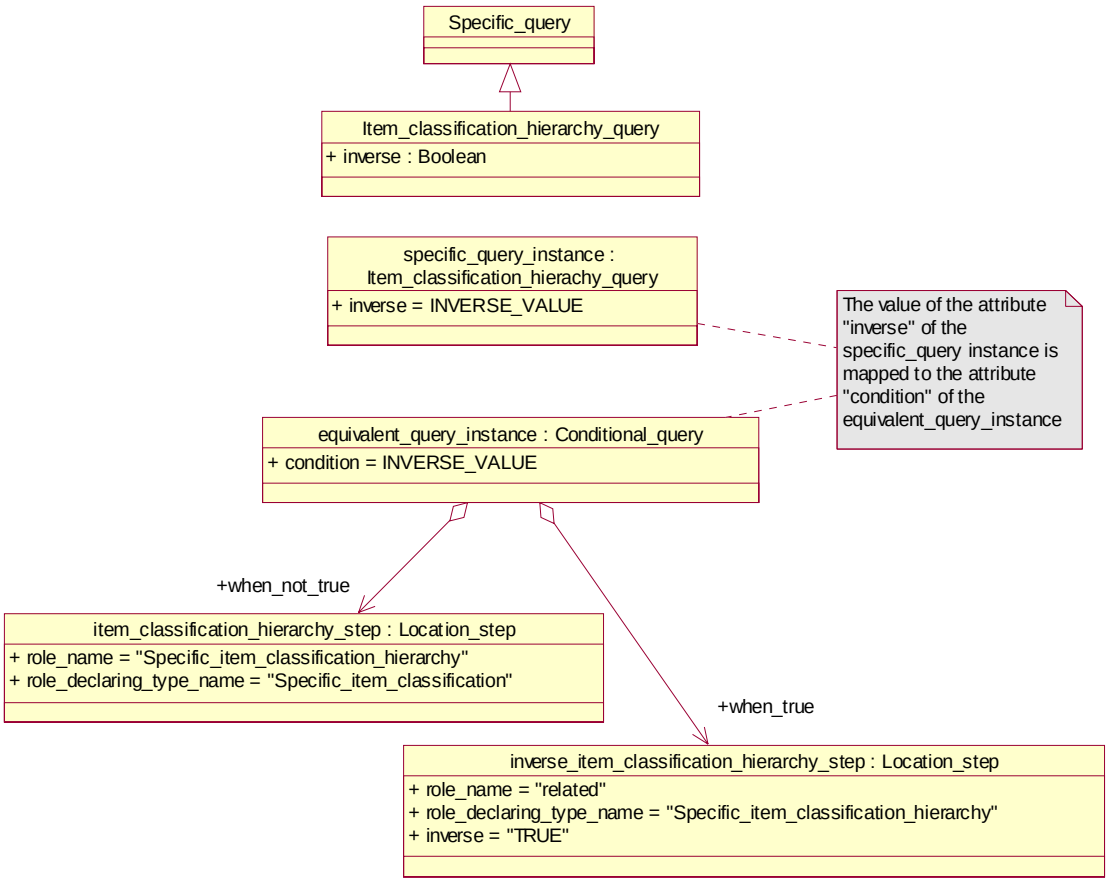


**Figure 9.61 - Definition, sample instance and equivalent Location_path instance of the Item_classification_hierarchy_query**

## 9.7.46 Item_definition_instance_relationship_query

The Item_definition_instance_relationship_query traverses from Design_discipline_item_definition objects to Item_instance objects or vice versa via Item_definition_instance_relationship objects

### *Base Class*

- Relationship_query

### *Parameters*

- Relationship_type_name: string [0..1]
  filters the Item_definition_instance_relationship objects by type, e.g. "Assembly_component_relationship" or "Next_higher_assembly".

- product_identification_uid: UID [0..1]

  identifies Product_identification object that represents a service specific algorithm to filter the Item_version objects.

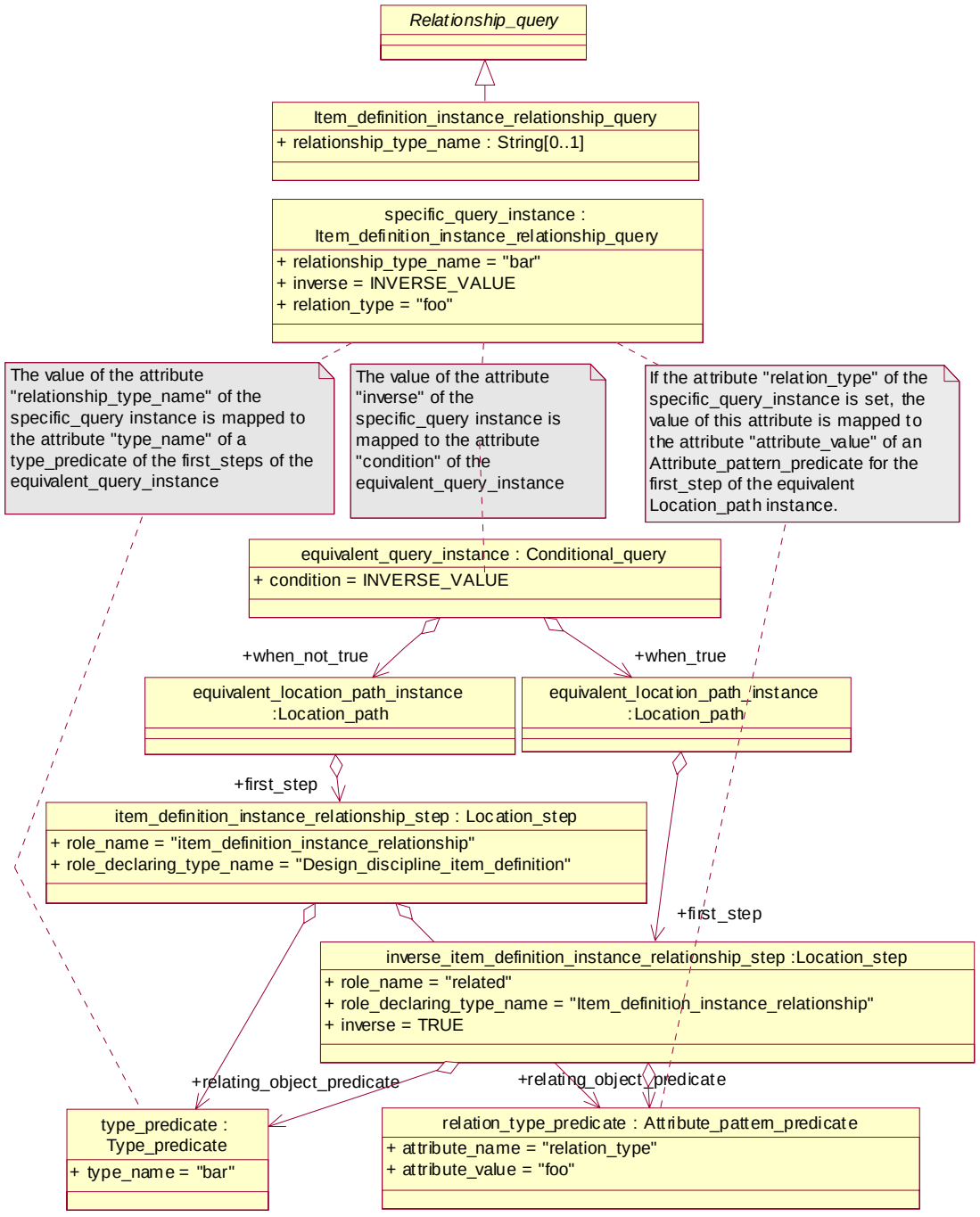


**Figure 9.62 - Definition, sample instance and equivalent Location_path instance of the Item_definition_instance_relationship_query**

### 9.7.47 Item_instance_query

The Item_instance_query traverses from Design_discipline_item_definition and Product_identification objects to Item_instance objects.

#### *Base Class*

- Query_with_relating_type_predicate

#### *Parameters*

- Id: string [0..1]
- Id_scope: string [0..1]
- Name: string [0..1]
- Name_language: language [0..1]

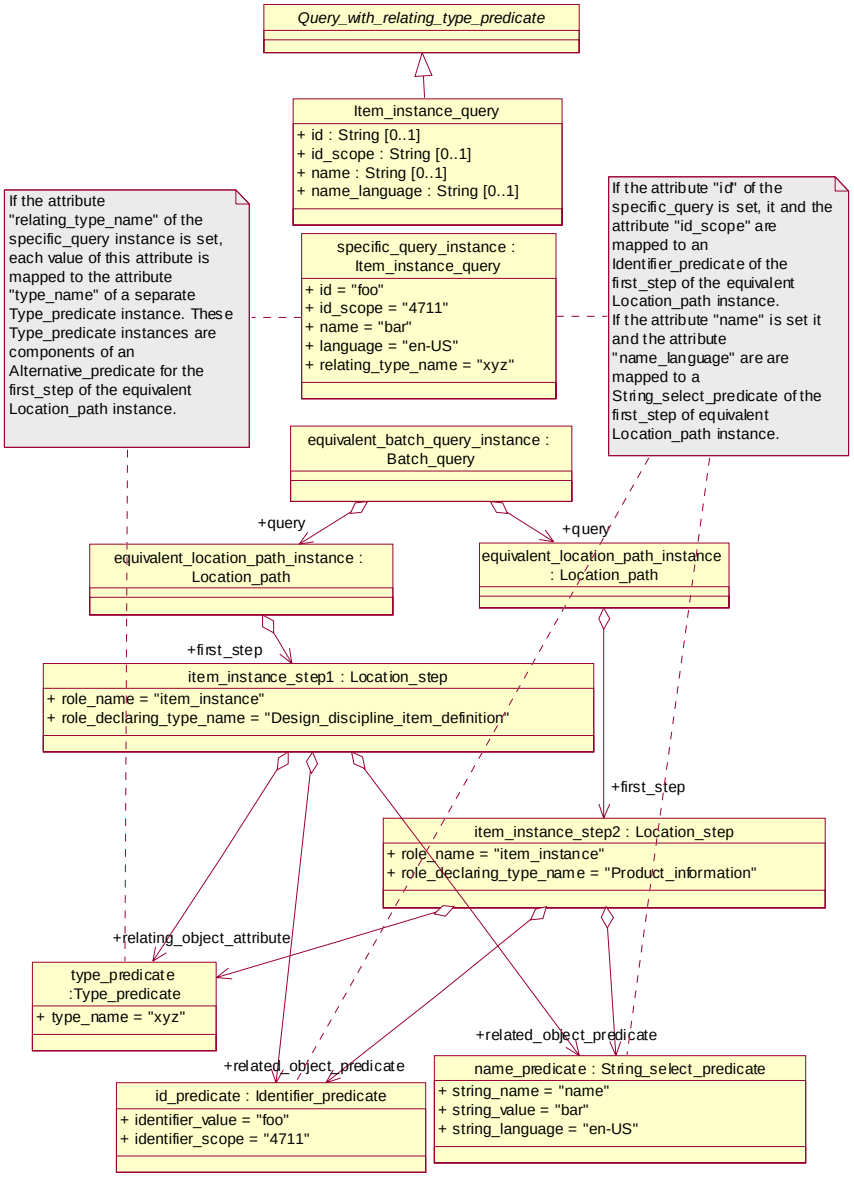


**Figure 9.63 - Definition, sample instance and equivalent Location_path instance of the Item_instance _query**

## 9.7.48 Item_query

The Item_query selects Item objects.

### *Parameters*

- id : String

- id_scope : String [0..1]
- name : String
- name_language : Language
- version_id : String [0..1]
- classification_name : String [0..1]

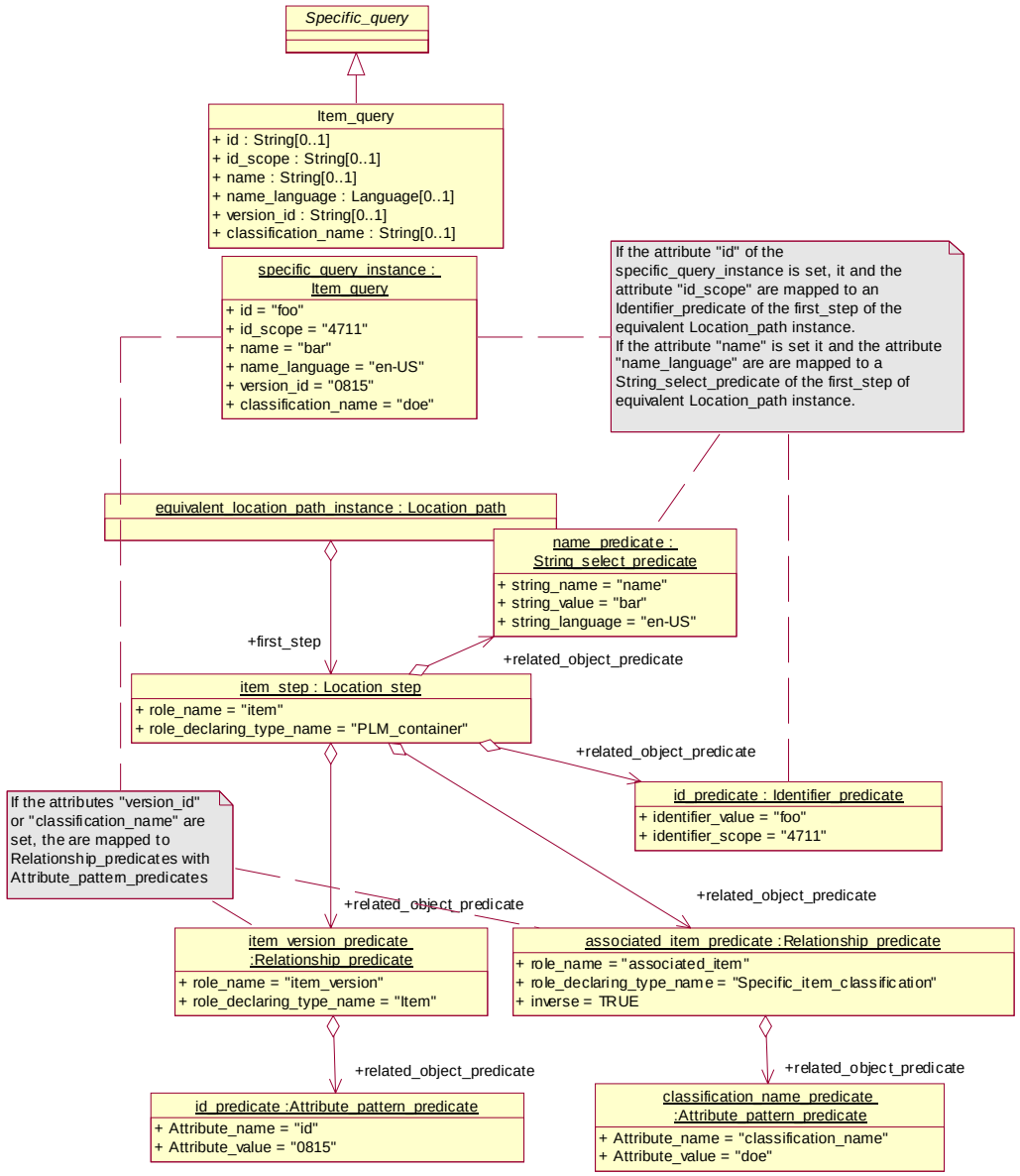


**Figure 9.64 - Definition, sample instance and equivalent Location_path instance of the Item_query**

### 9.7.49 Item_use_query

The Item_use_query traverses those assemblies from Design_discipline_item_definition objects where the Design_discipline_item_definition objects are used as components.

#### *Parameters*

- none

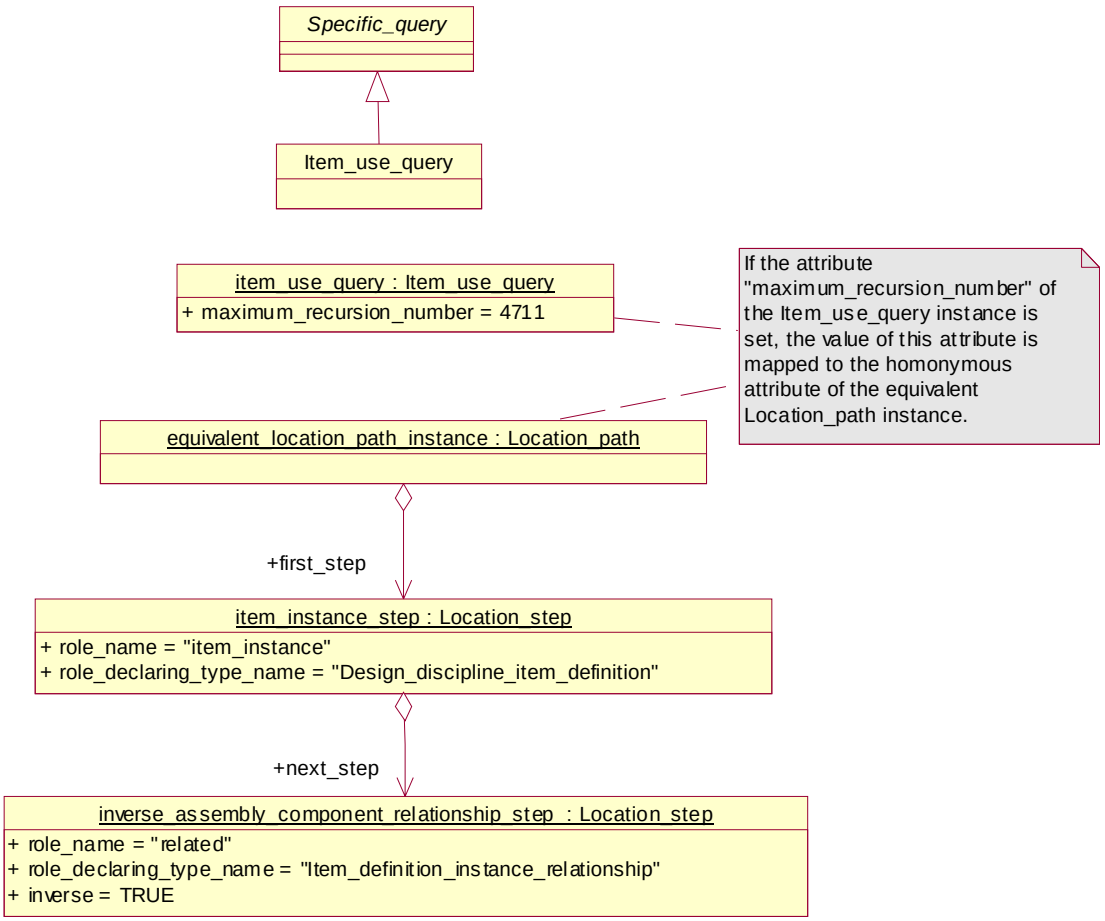


**Figure 9.65 - Definition, sample instance and equivalent Location_path instance of the Item_use_query**

### 9.7.50 Item_version_query

The Item_version_query traverses Item_version objects from Item objects.

#### *Parameters*

- id : String [0..1]

- id_scope : String [0..1]
- product_identification_uid: UID [0..1]
  identifies Product_identification object that represents a service specific algorithm to filter the Item_version objects.

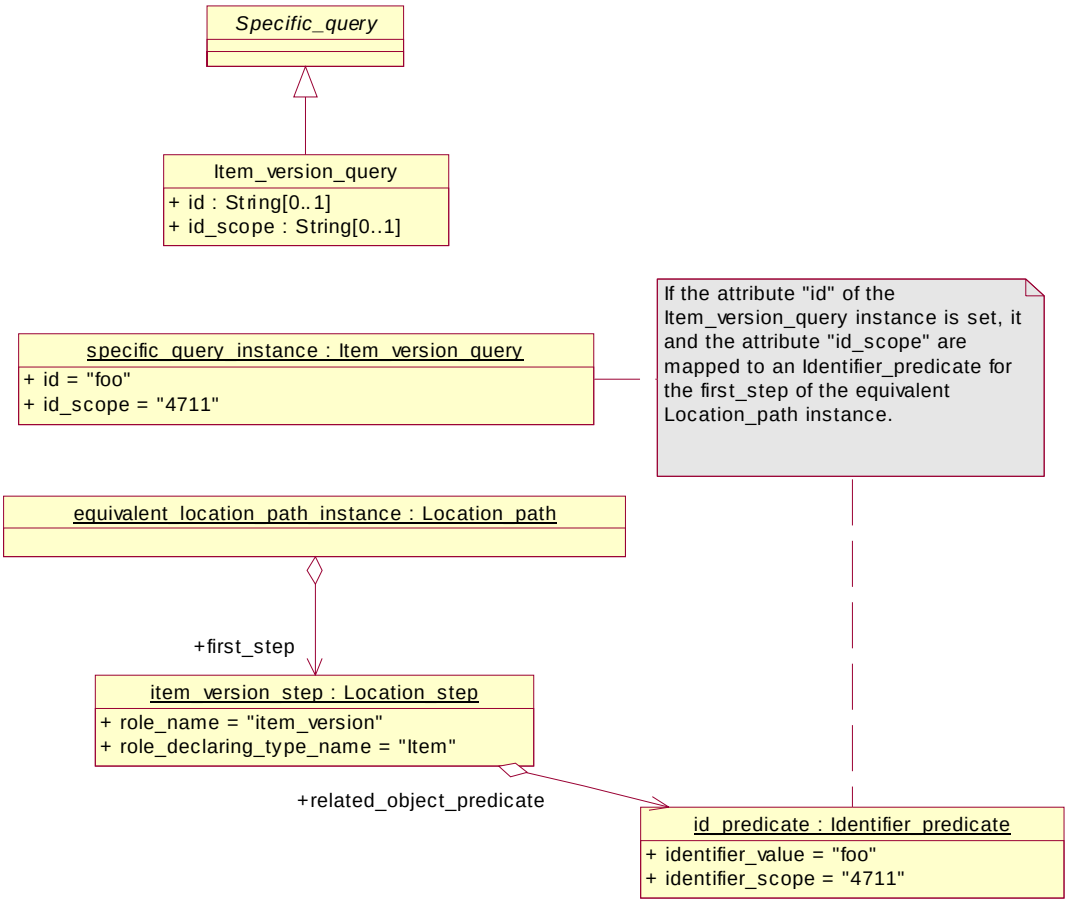


**Figure 9.66 - Definition, sample instance and equivalent Location_path instance of the Item_version_query**

### 9.7.51 Item_version_relationship_query

The Item_version_relationship_query traverses from Item_version objects via Item_version_relationship objects to Item_version objects.

#### *Parameters*

- relation_type : String [0..1] the relation_type attribute of the queried relationships
- inverse : Boolean [0..1]

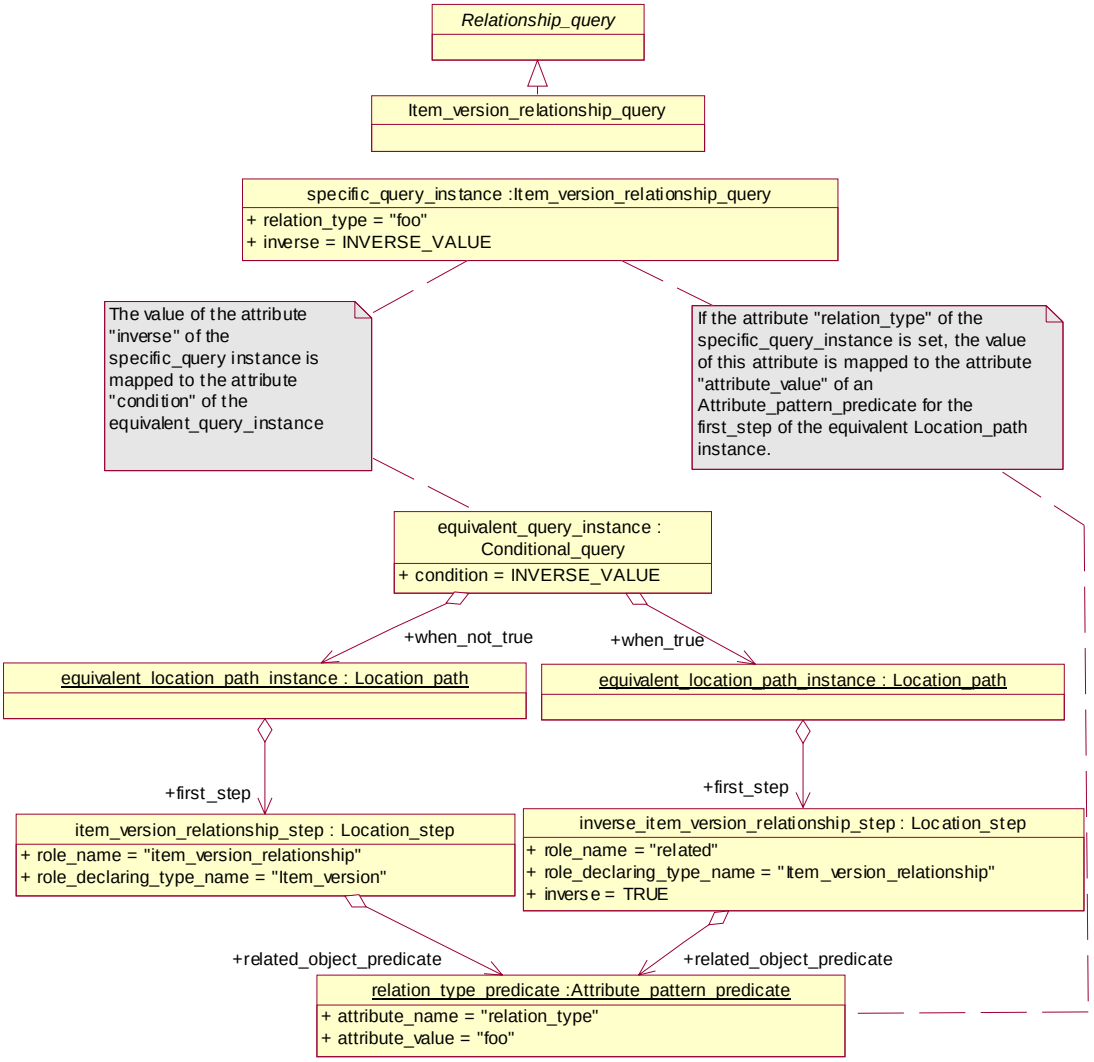


**Figure 9.67 - Definition, sample instance and equivalent Location_path instance of the Item_version_relationship_query**

## 9.7.52 Object_by_uid_query

The Object_by_uid_query selects an object by its uid.

### *Parameters*

- uid : UID
- opaque_server_data : Byte[0..*]

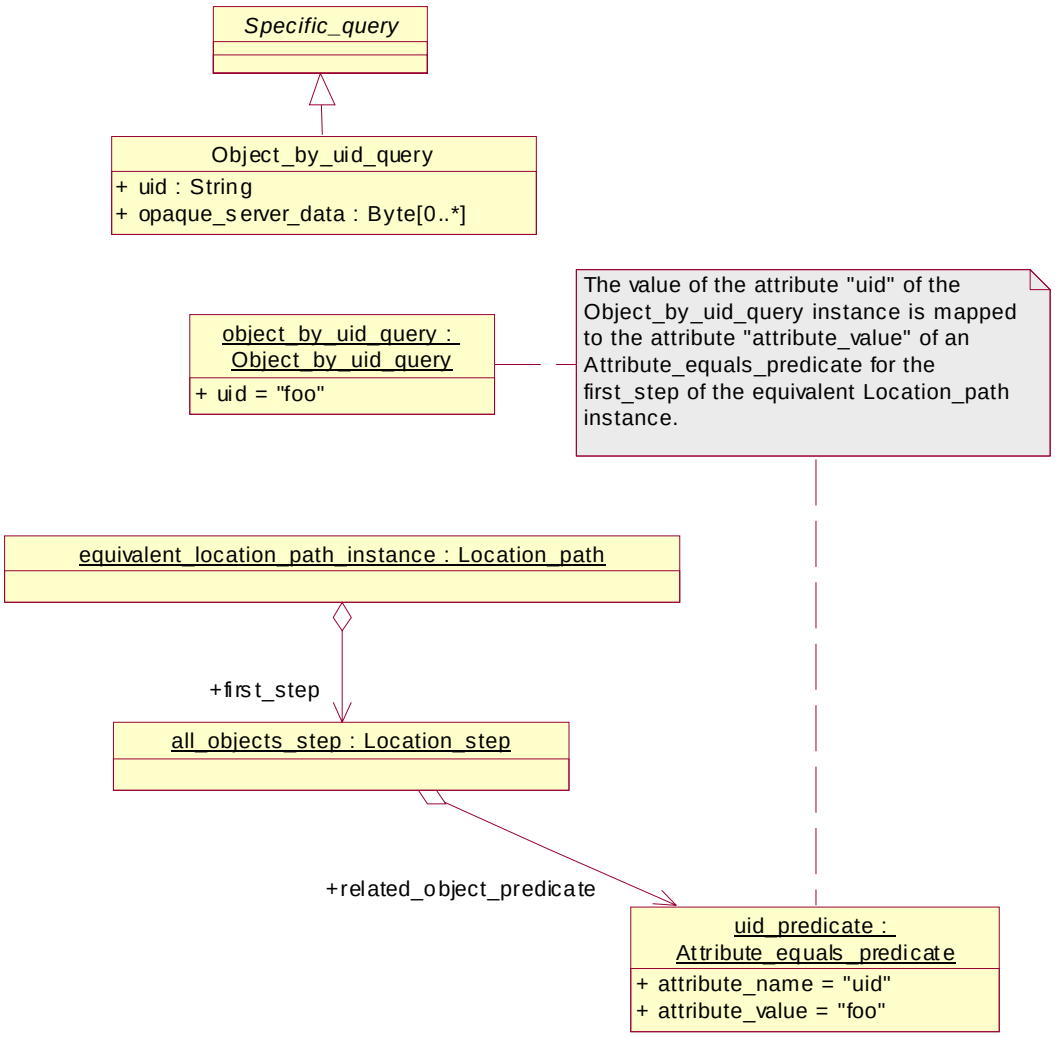


**Figure 9.68 - Definition, sample instance and equivalent Location_path instance of the Object_by_uid_query**

### 9.7.53 Objects_by_uids_query

The Objects_by_uids_query selects a set of objects by its uids.

#### *Parameters*

- uids : UID[1..*]

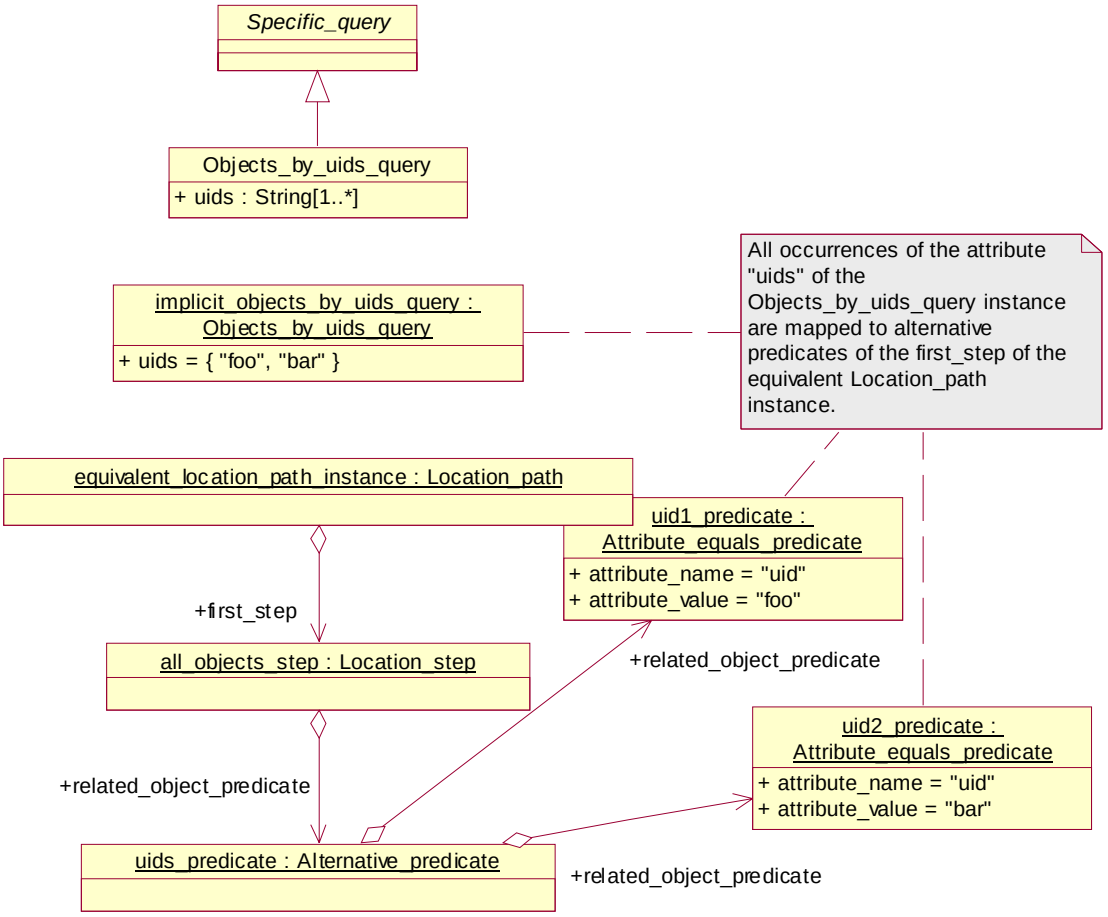


**Figure 9.69 - Definition, instance and equivalent explicit Location_path instance of the Objects_by_uids_query**

### 9.7.54 Organization_query

The Organization_query selects Organization objects.

#### *Parameters*

- id : String [0..1] the id of the Organization for which the information is queried
- Id_scope : String [0..1]
- organization_name : String [0..1]
- organization_type : String [0..1]

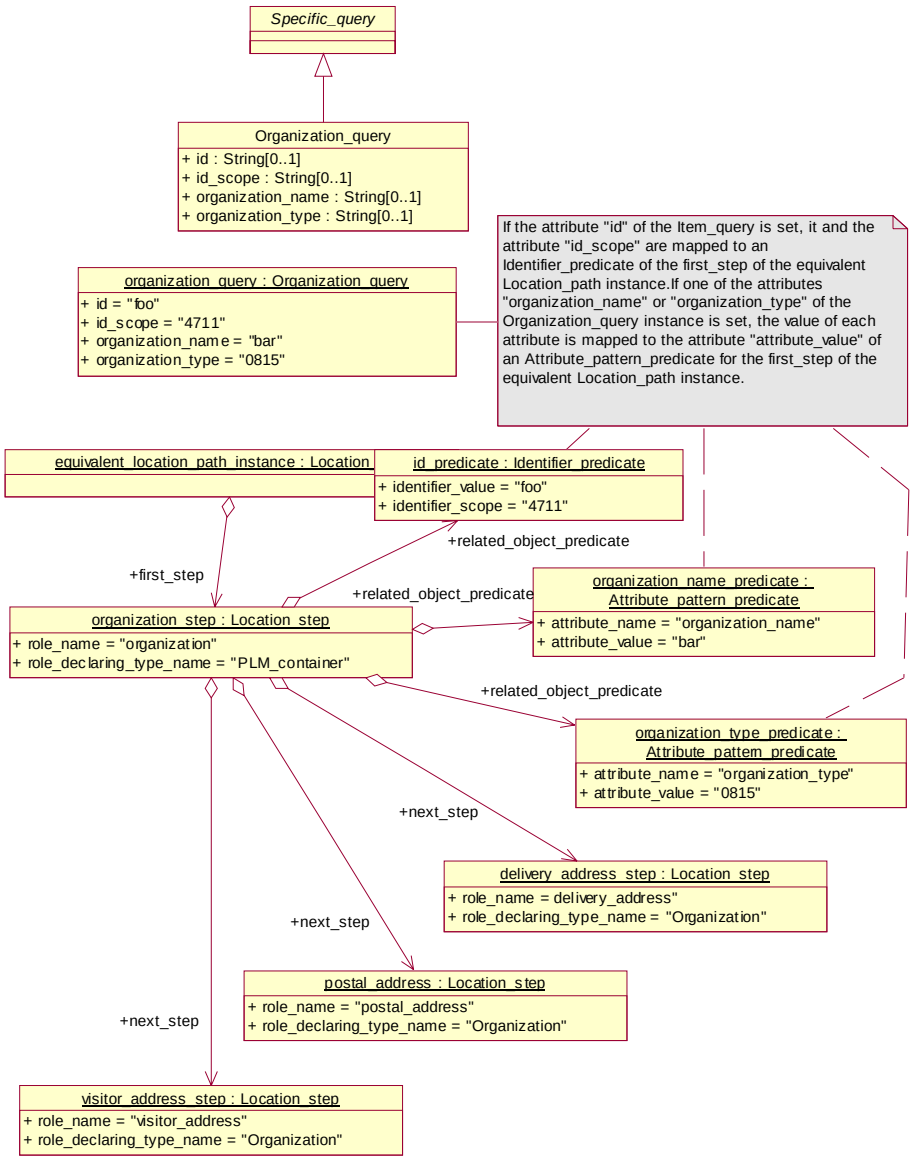


**Figure 9.70 - Definition, sample instance and equivalent Location_path instance of the Organization_query**

## 9.7.55 Organization_relationship_query

The Organization_relationship_query traverses from Organization objects via Organization_relationship objects to Organization objects.

***Parameters***

- relation_type : String [0..1]
- inverse : Boolean [0..1]

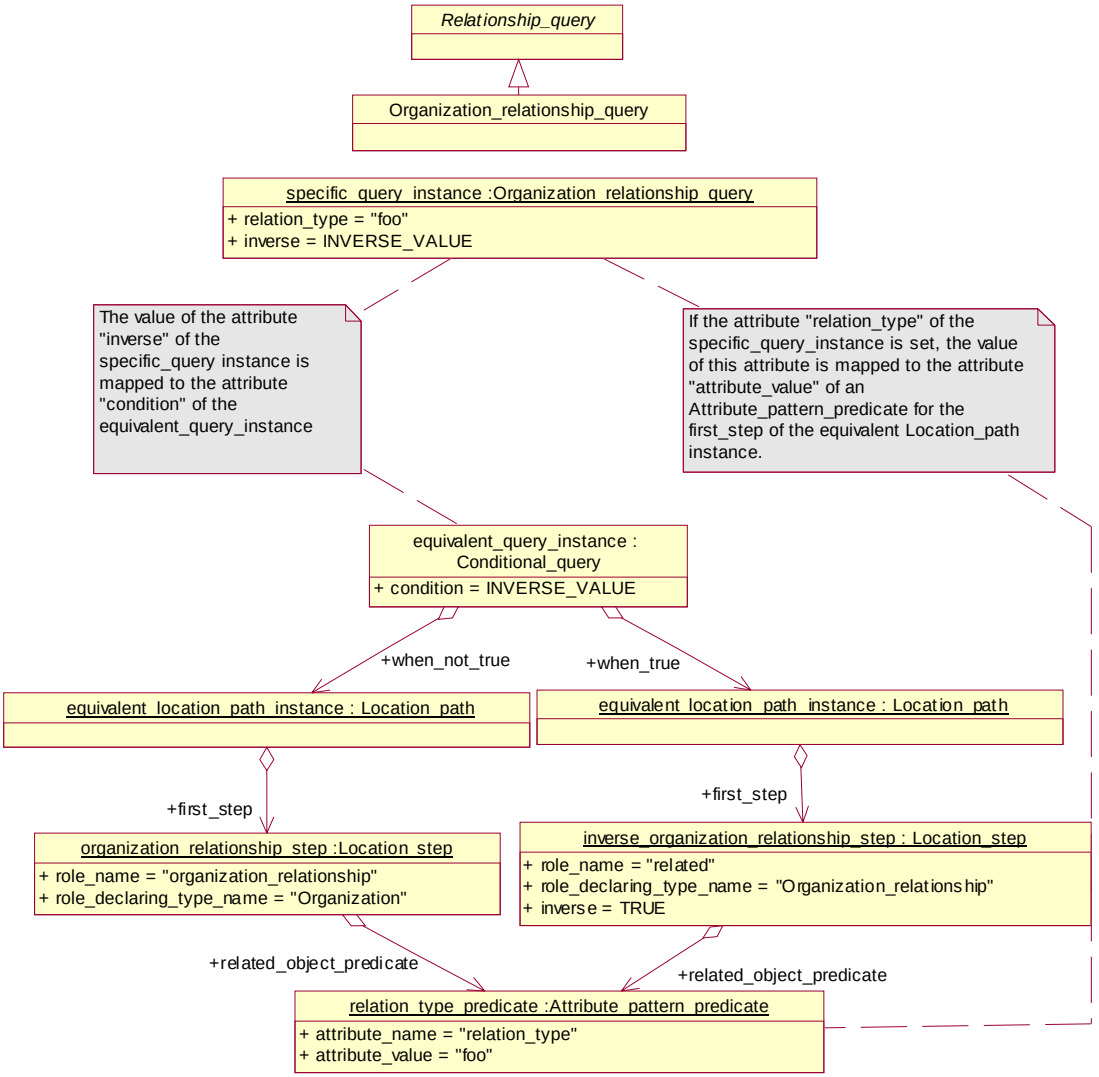


**Figure 9.71 - Definition, sample instance and equivalent Location_path instance of the Organization_relationship_query**

## 9.7.56 Person_in_organization_query

The Person_in_organization_ query traverses from Person objects via Person_in_organization objects to Organization objects.

***Parameters***

- person_name : String [0..1]
- id: String [0..1]
- role: String [0..1]
- organization_id : String [0..1]

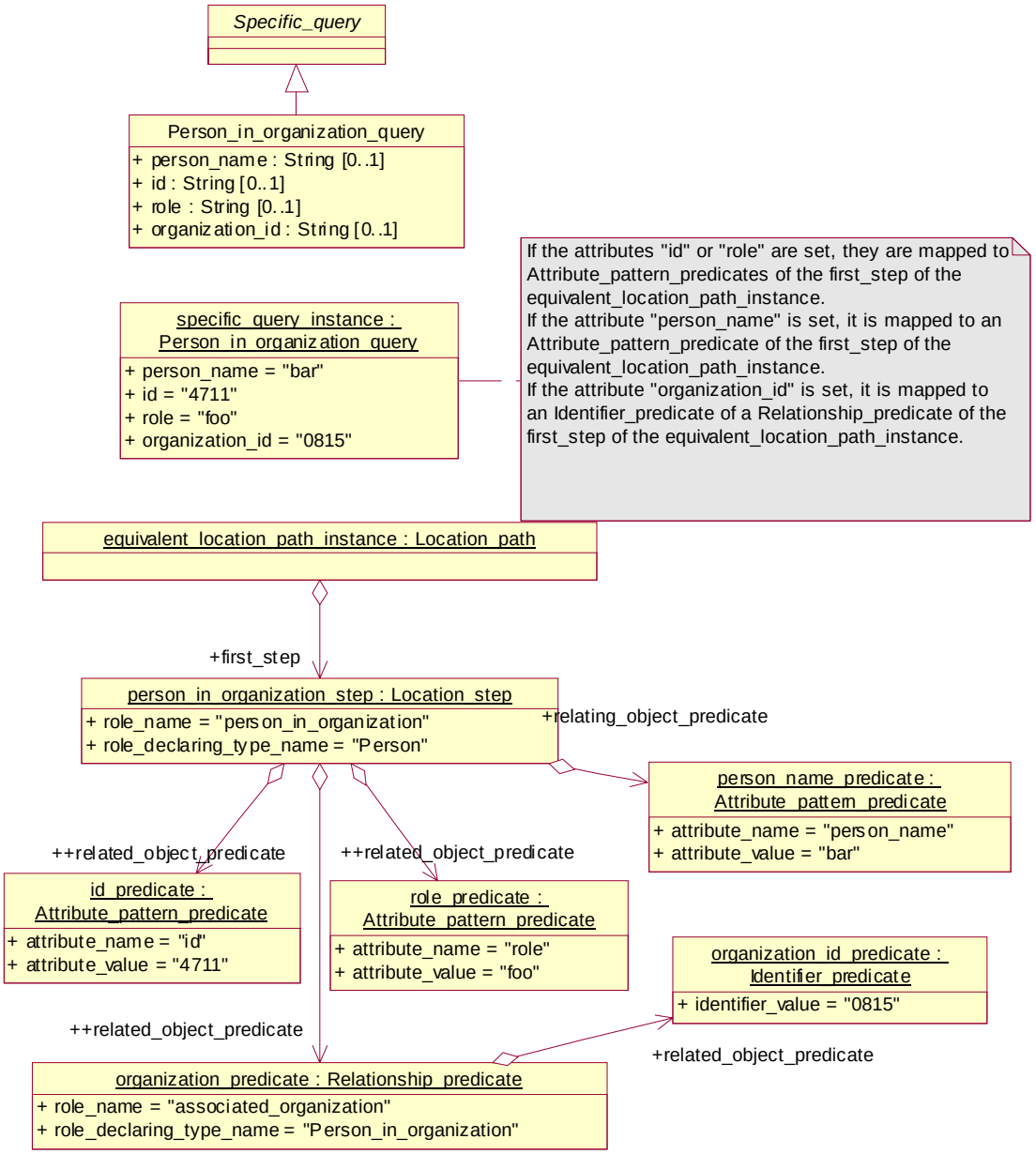


**Figure 9.72 - Definition, sample instance and equivalent Location_path instance of the Person_in_organization_query**

### 9.7.57 Person_in_organization_relationship_query

The Person_in_organization_relationship_query traverses from Person_in_organization objects via Person_in_organization_relationship objects to Person_in_organization objects.

***Parameters***

- relation_type : String [0..1]
- inverse : Boolean [0..1]

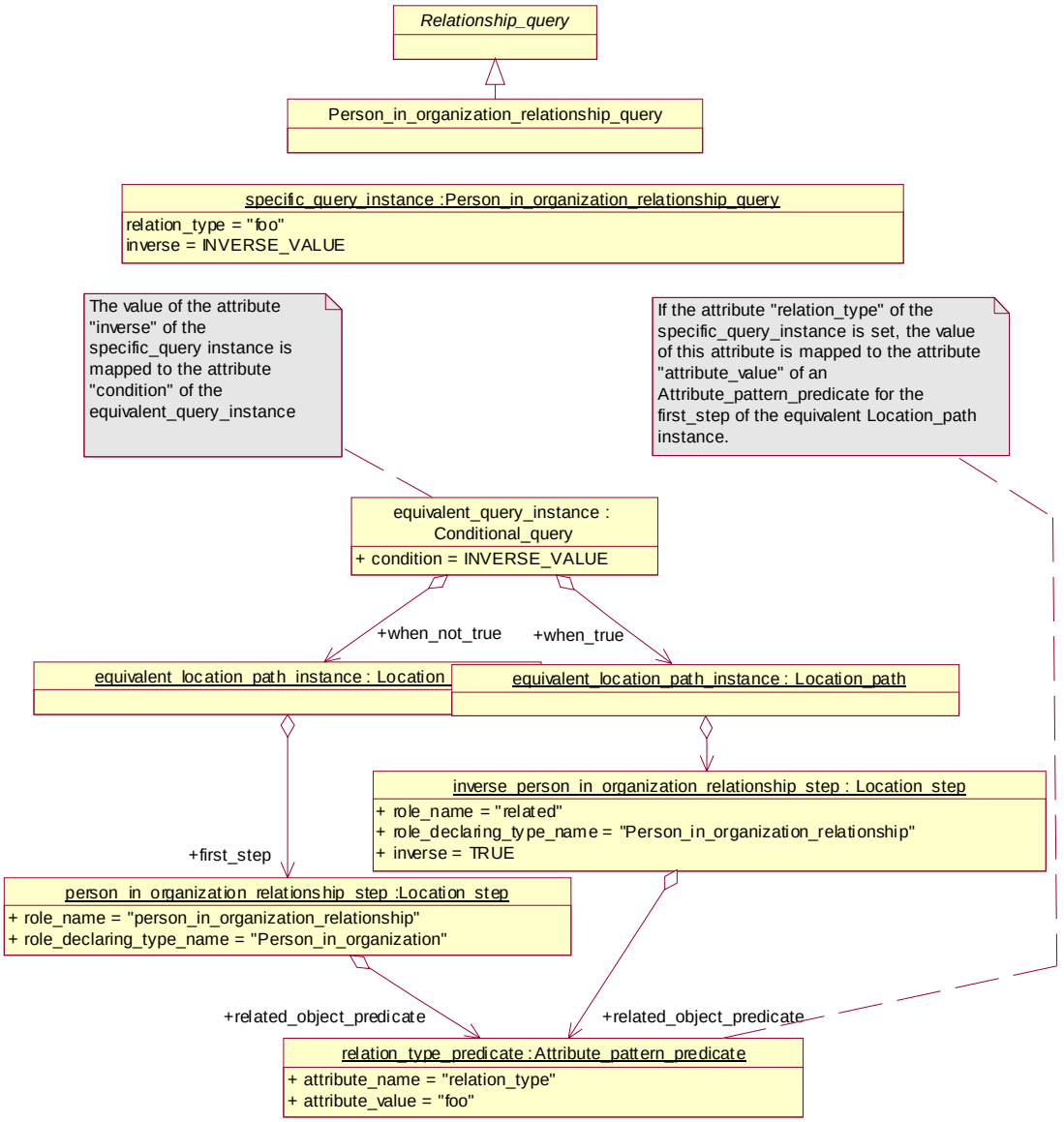


**Figure 9.73 - Definition, sample instance and equivalent Location_path instance of the Person_in_organization_relationship_query**

### 9.7.58 Person_organization_assignment_query

The Person_organization_assignment_query traverses from Person_organization_select objects via Person_organization_assignment objects to Date_time_person_organization_element_select objects.

***Parameters***

- role : String [0..1]
- related_type_name : String [0..*]

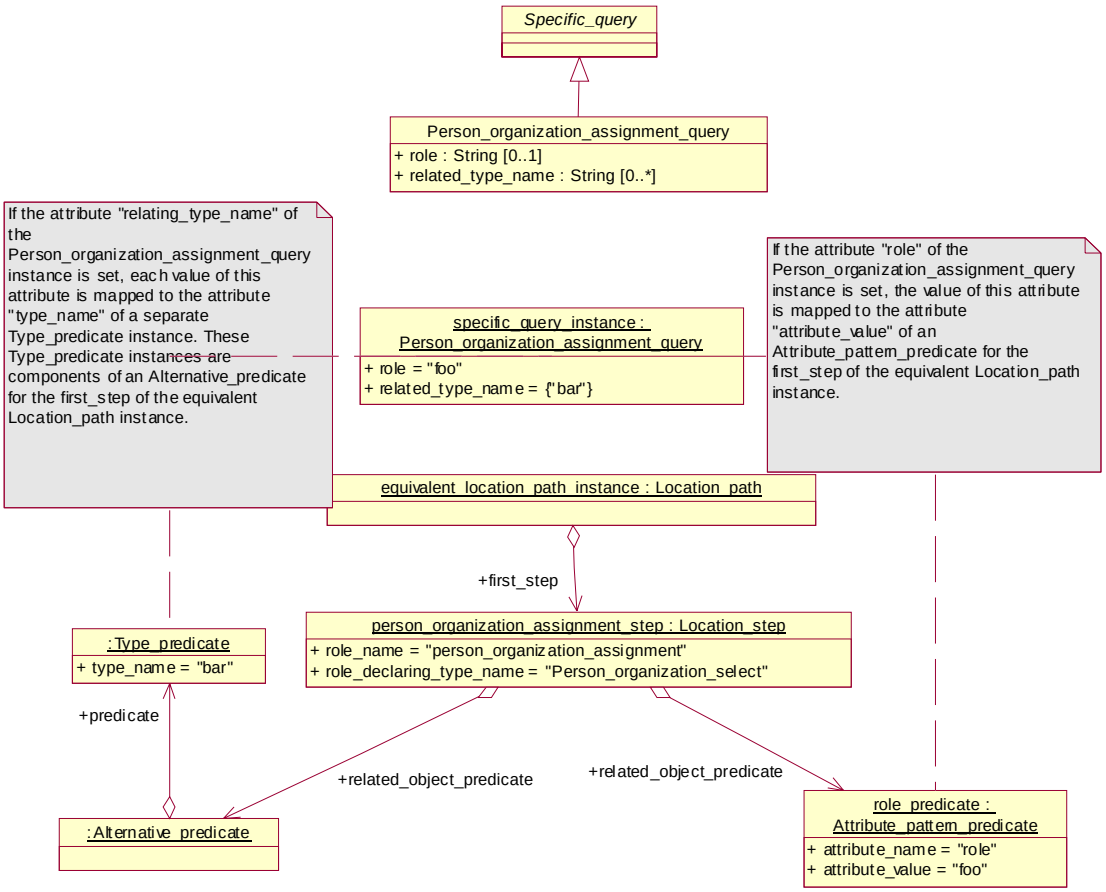


**Figure 9.74 - Definition, sample instance and equivalent Location_path instance of the Person_organization_assignment_query**

### 9.7.59 Person_query

The Person_query selects Person objects.

### *Parameters*

- person_name : String [0..1]
- id: String [0..1]
- role: String [0..1]
- organization_id : String [0..1]

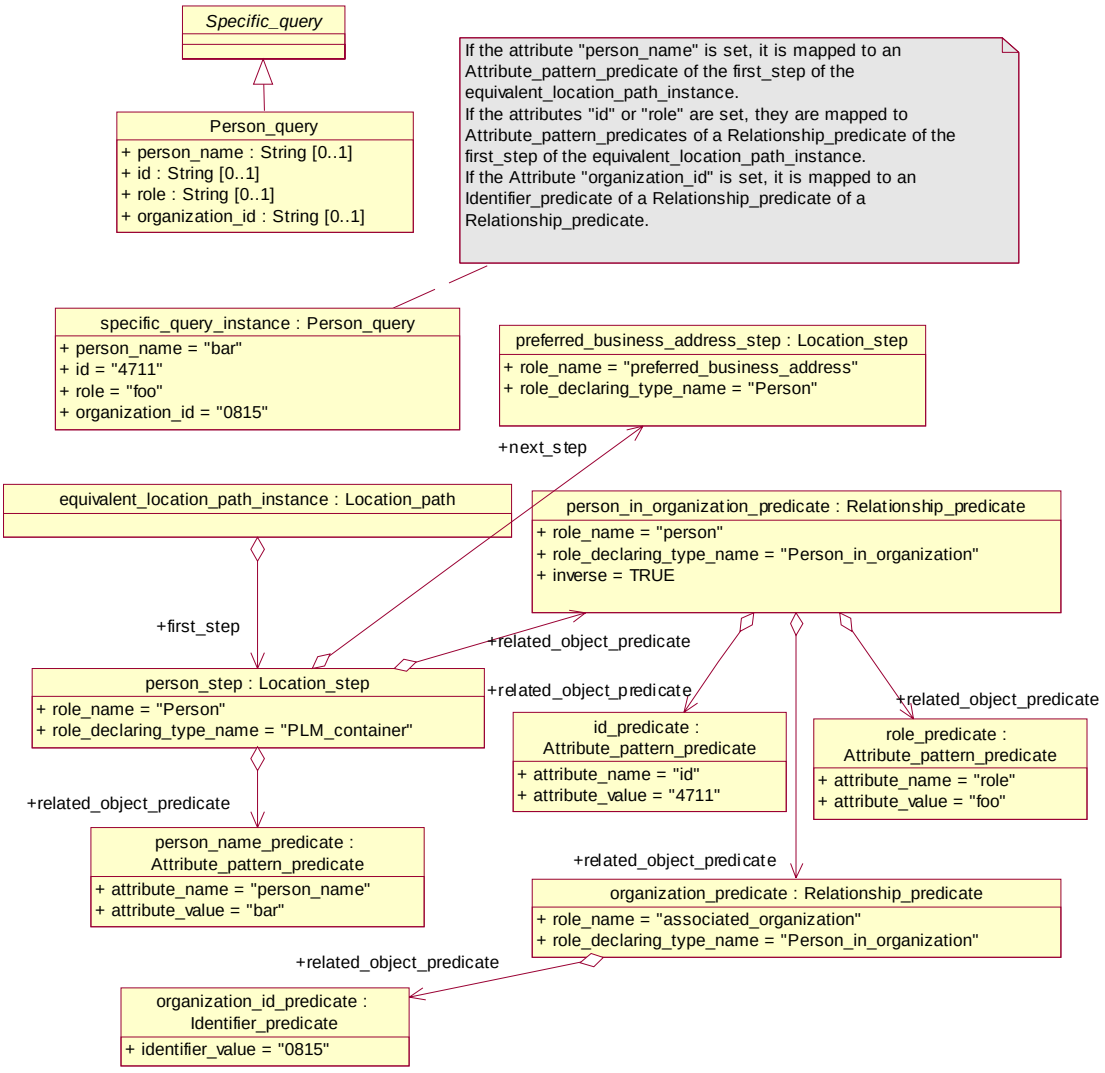


**Figure 9.75 - Definition, sample instance and equivalent Location_path instance of the Person_query**

### 9.7.60 Product_class_query

The Product_class_query selects Product_class objects.

***Parameters***

- id : String
- id_scope : String [0..1]
- name : String
- name_language : Language

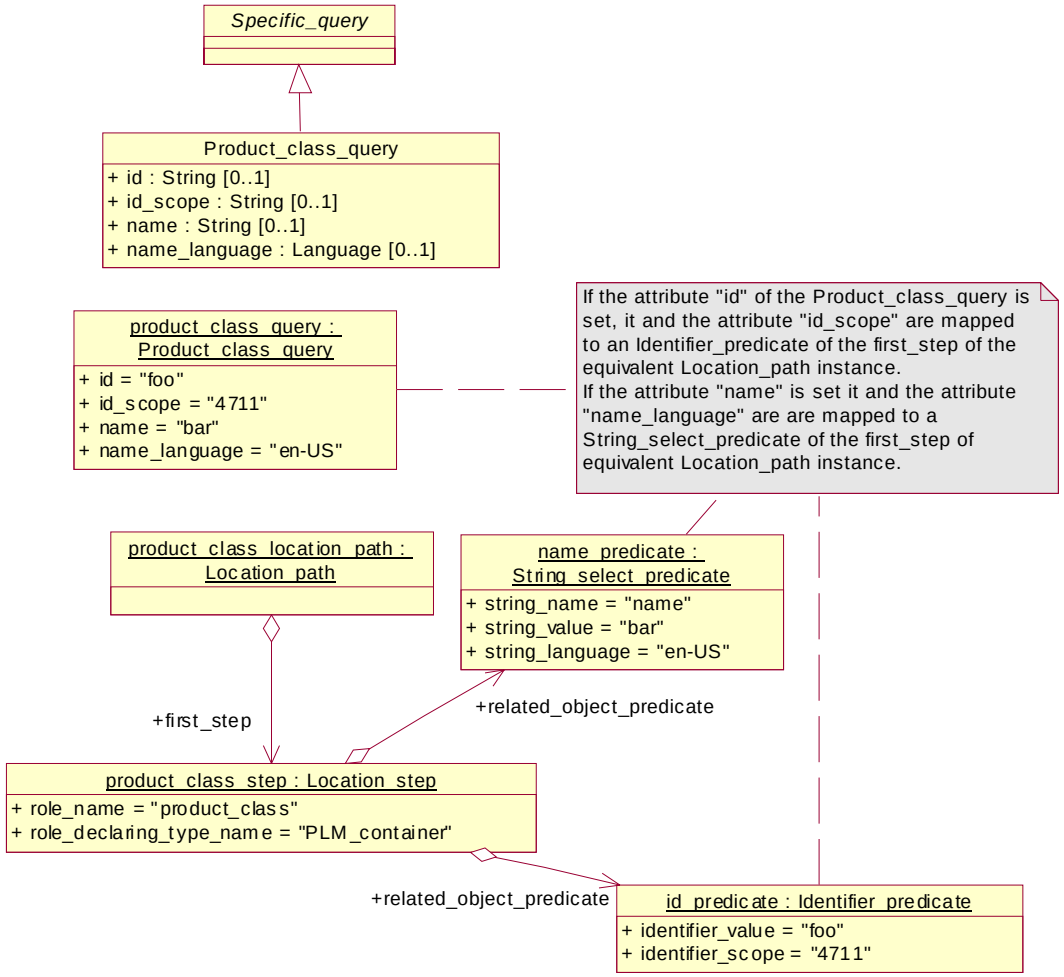


**Figure 9.76 - Definition, sample instance and equivalent Location_path instance of the Product_class_query**

### 9.7.61 Product_identification_query

The Product_identification_query selects Product_identification objects..

#### *Base Class*

- Specific_query

#### *Parameters*

- id: string [0..1]
- id_scope: string [0..1]
- name: string [0..1]
- name_language: language [0..1]
- version_id: string [0..1]

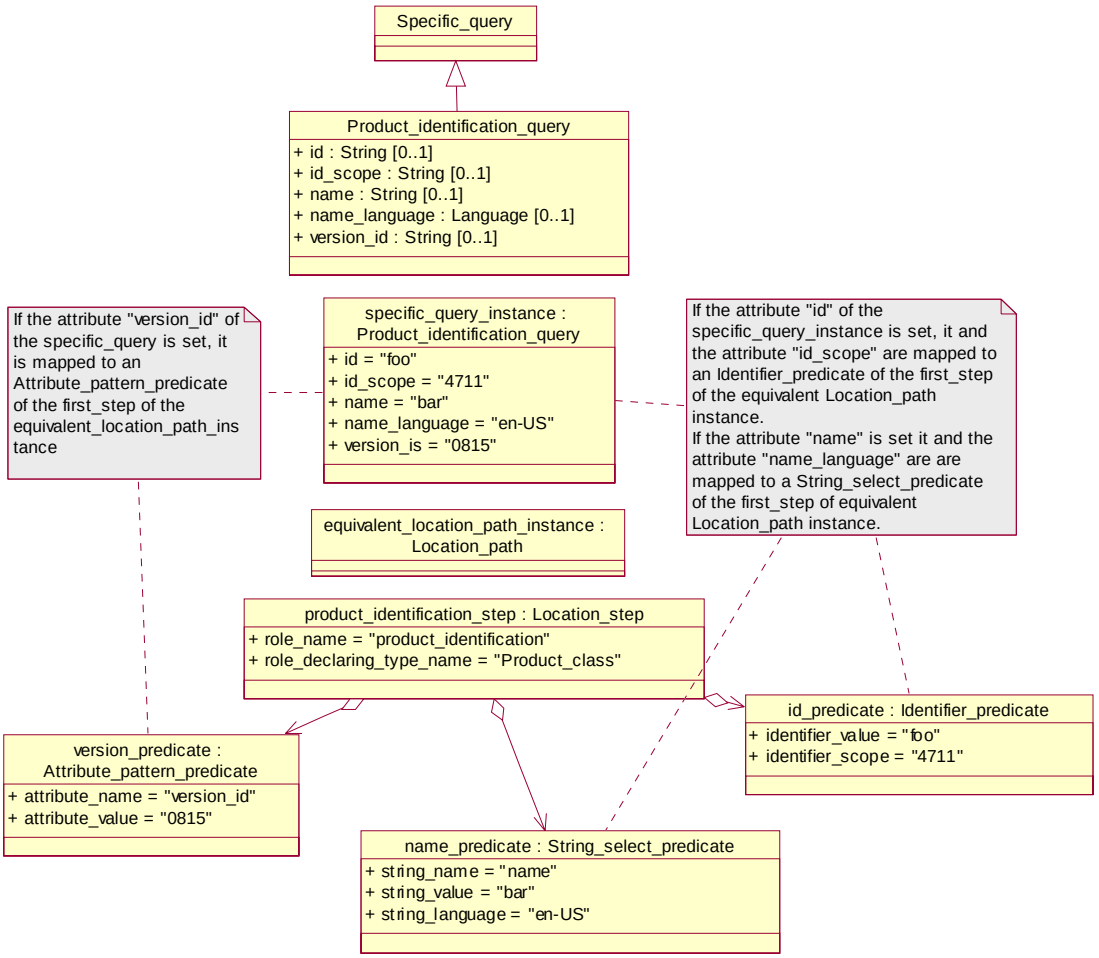


**Figure 9.77 - Definition, sample instance and equivalent Location_path instance of the Product_identification_query**

## 9.7.62 Product_structure_query

The Product_structure_query traverses Product_structure_relationship objects from Complex_product objects.

### *Parameters*

- relation_type : String[0..1]

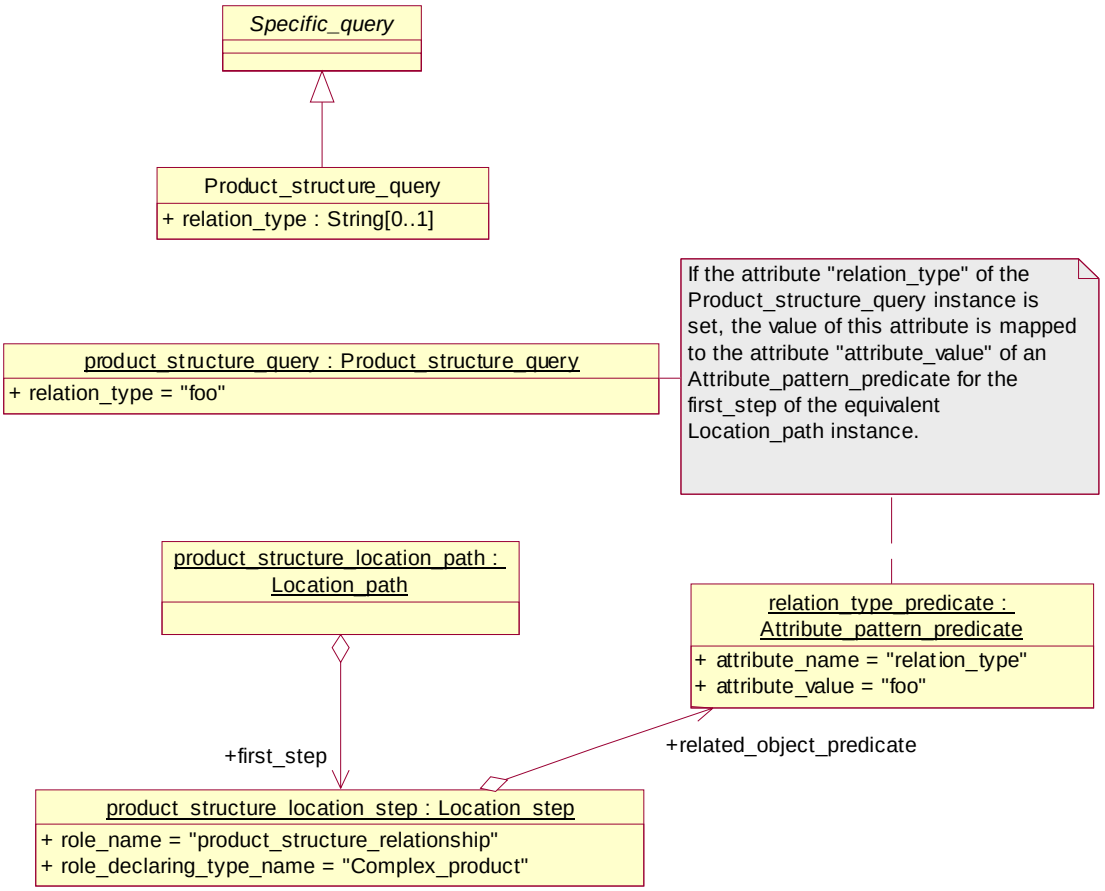


**Figure 9.78 - Definition, sample instance and equivalent Location_path instance of the Product_structure_query**

### 9.7.63 Project_assignment_query

The Project_assignment_query traverses from Project objects via Project_assignment objects to Project_information_select objects.

#### *Parameters*

- role : String [0..1]
- related_type_name : String [0..*]

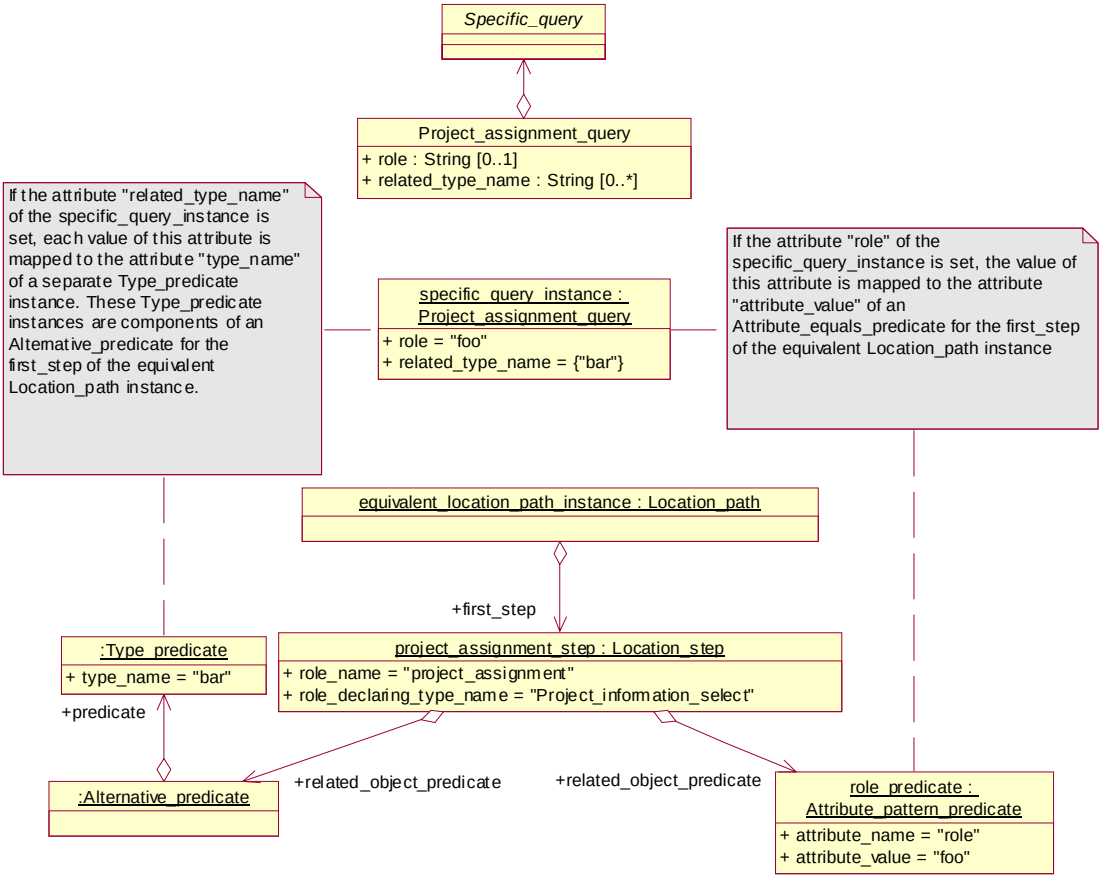


**Figure 9.79 - Definition, sample instance and equivalent Location_path instance of the Project_assignment_query**

### 9.7.64 Project_query

The Project_query selects Project objects.

#### *Base Class*

- Specific_query

#### *Parameters*

- id: string [0..1]
- id_scope: string [0..1]
- name: string [0..1]
- name_language: language [0..1]

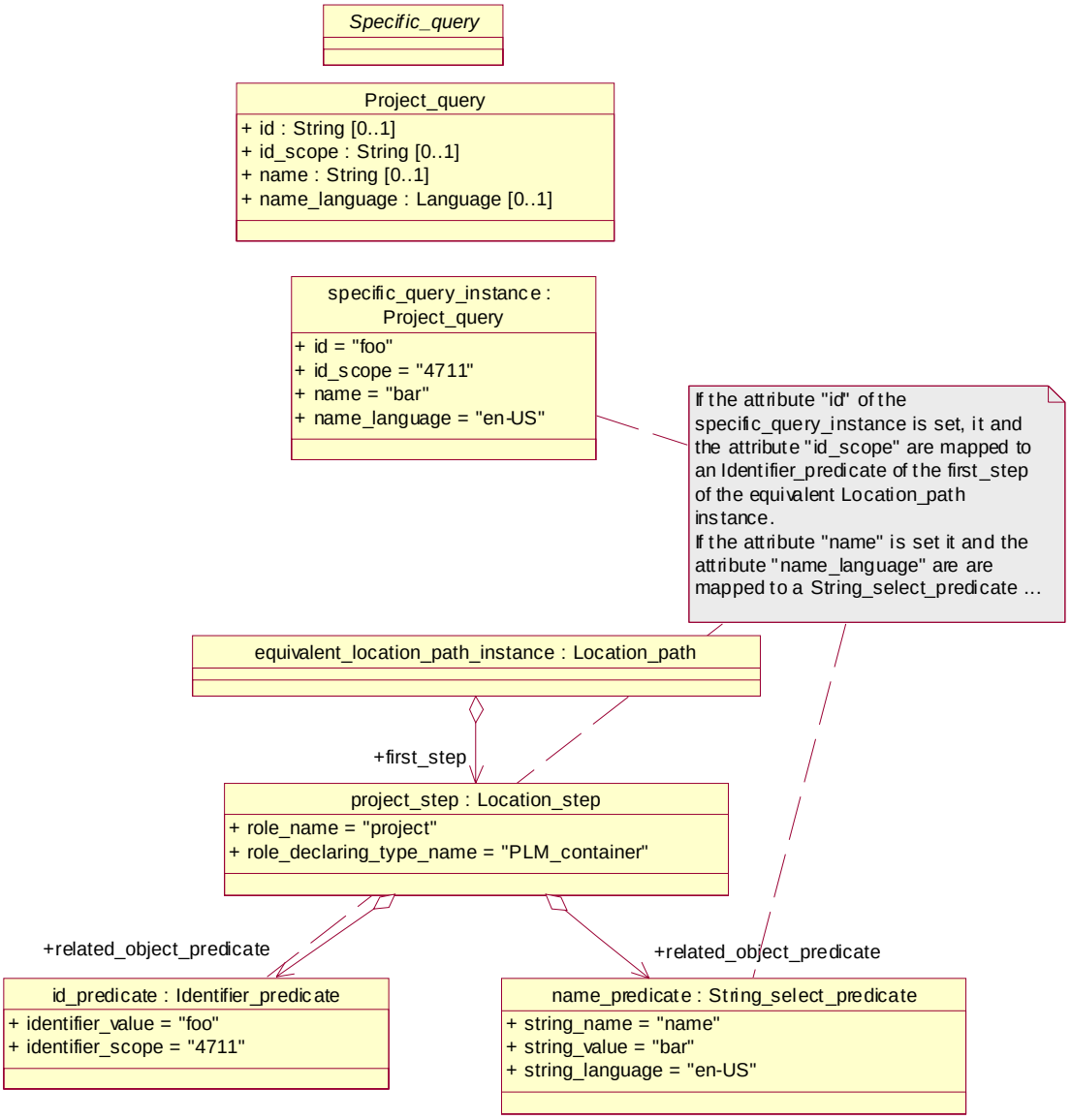


**Figure 9.80 - Definition, sample instance and equivalent Location_path instance of the Project_ query**

## 9.7.65 Simple_property_value_query

The Simple_property_query traverses from Simple_property_select objects via Simple_property_association objects to Simple_property_value objects.

### *Parameters*

- relating_type_name : String [0..*]
- value_name : String [0..1]

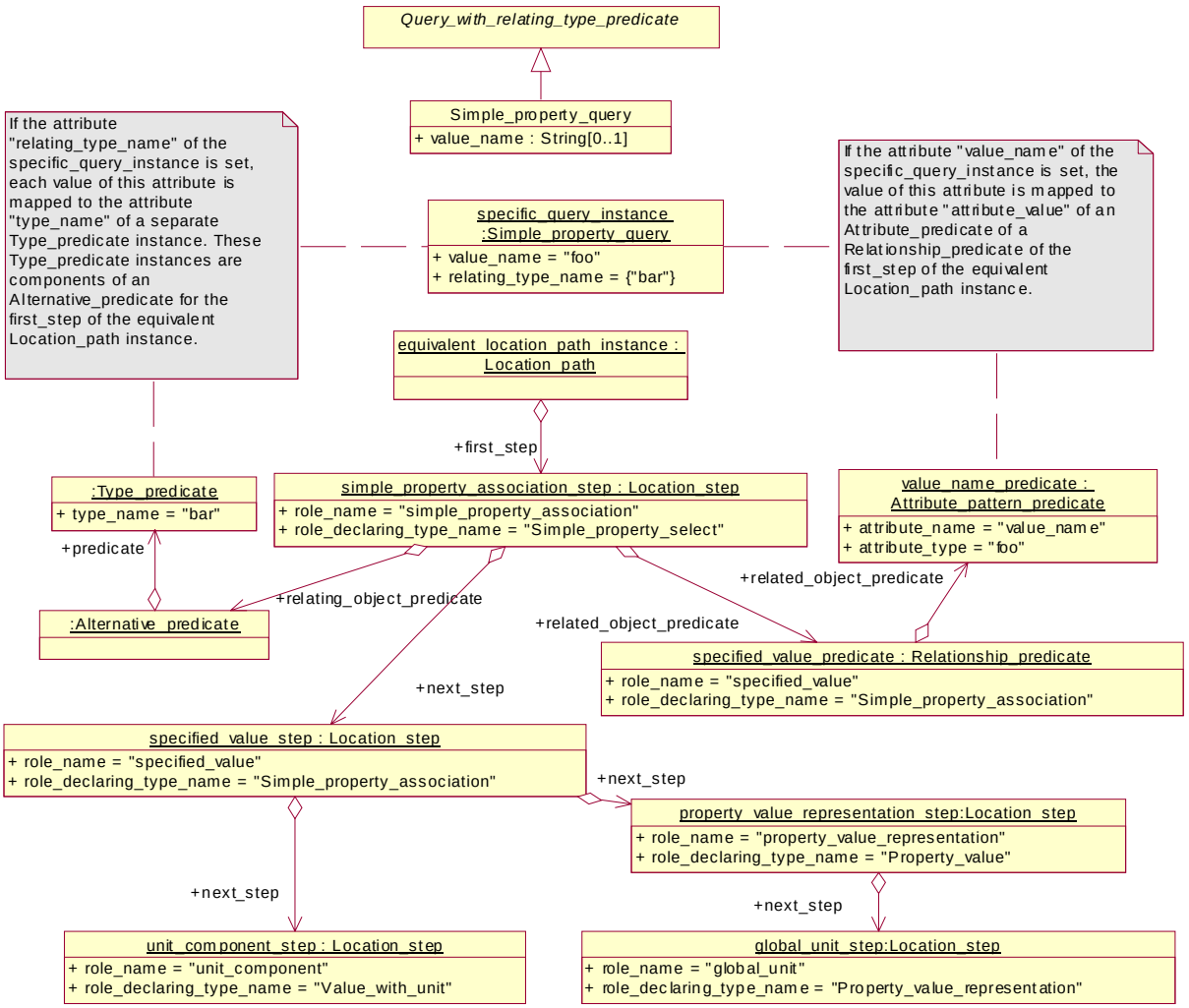


**Figure 9.81 - Definition, sample instance and equivalent Location_path instance of the Simple_property_value_query**

## 9.7.66 Work_order_query

The Work_order_query selects Work_order objects.

### *Base Class*

- Specific_query

### *Parameters*

- id: string [0..1]
- id_scope: string [0..1]

- work_order_type: string [0..1]
- version_id: string [0..1]
- classification_role: string [0..1]
- classification_id: string [0..1]
- description: string [0..1]
- description_language: string [0..1]

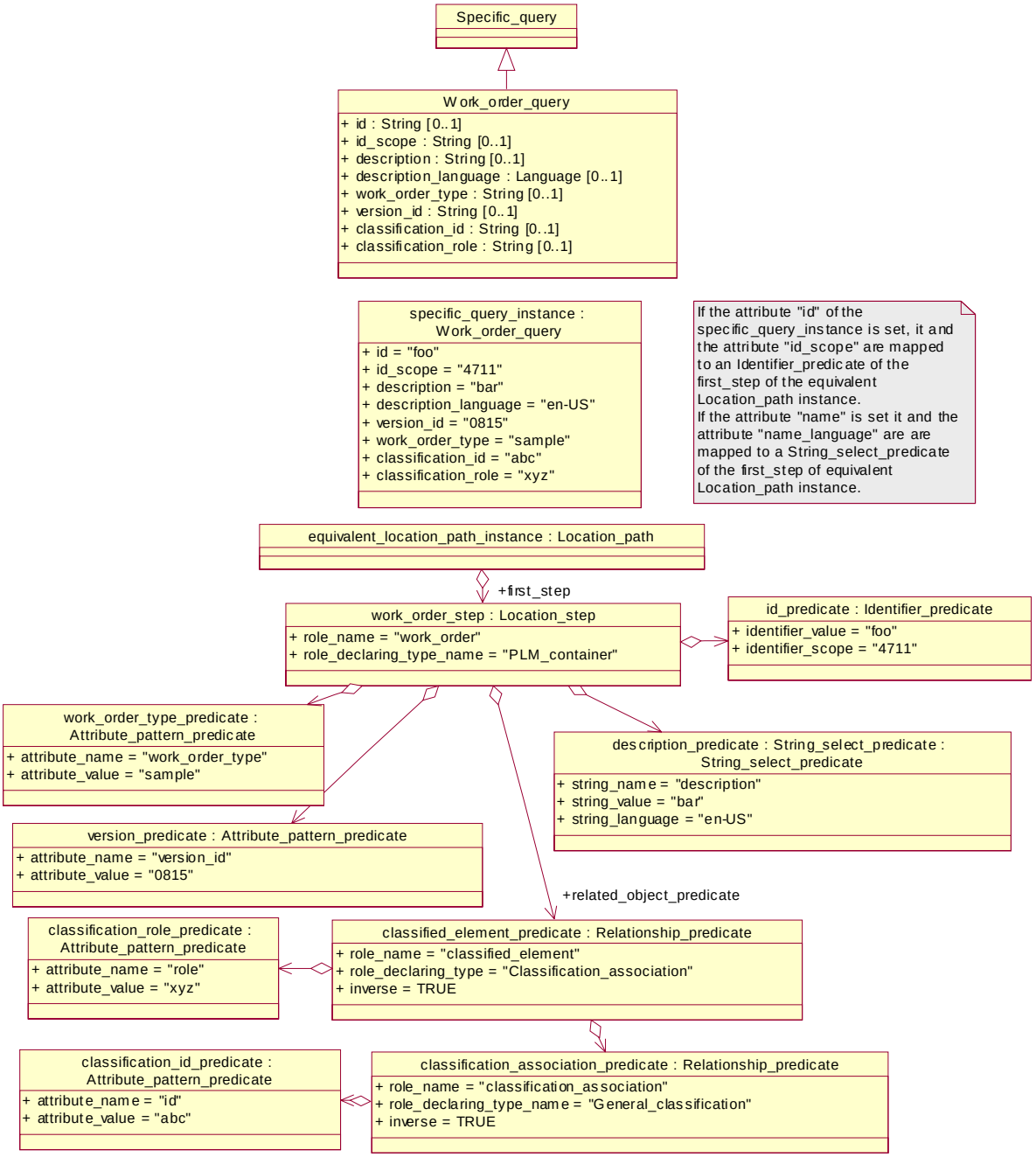


**Figure 9.82 - Definition, sample instance and equivalent Location_path instance of the Work_order_query**

### 9.7.67 Work_order_is_controlling_query

The Work_order_is_controlling_query traverses from Work_order objects to the Activity objects which are referenced by the is_controlling reference.

#### *Base Class*

- Specific_query

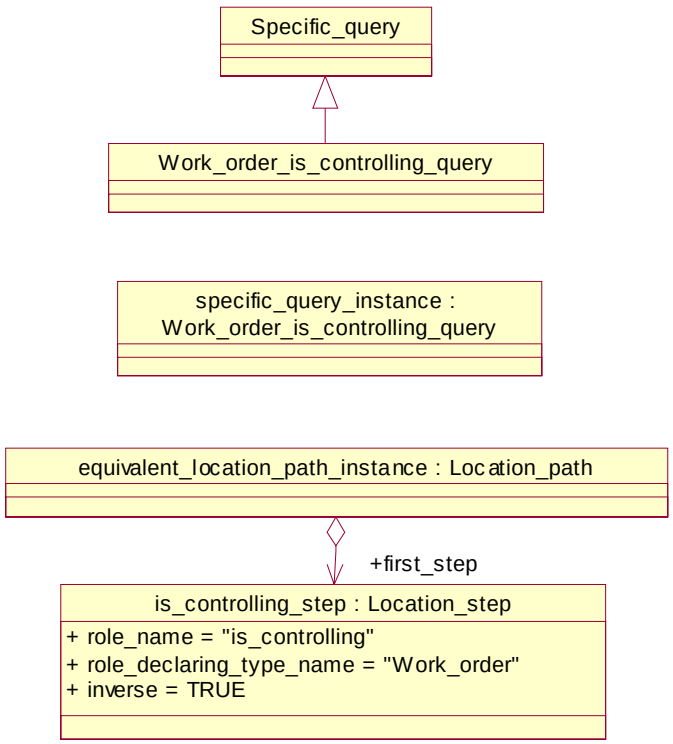


**Figure 9.83 - Definition, sample instance and equivalent Location_path instance of the Work_order_is_controlling_query**

### 9.7.68 Work_request_activity_query

The Work_request_activity_query traverses from Work_request objects to Activity objects.

#### *Parameters*

- none

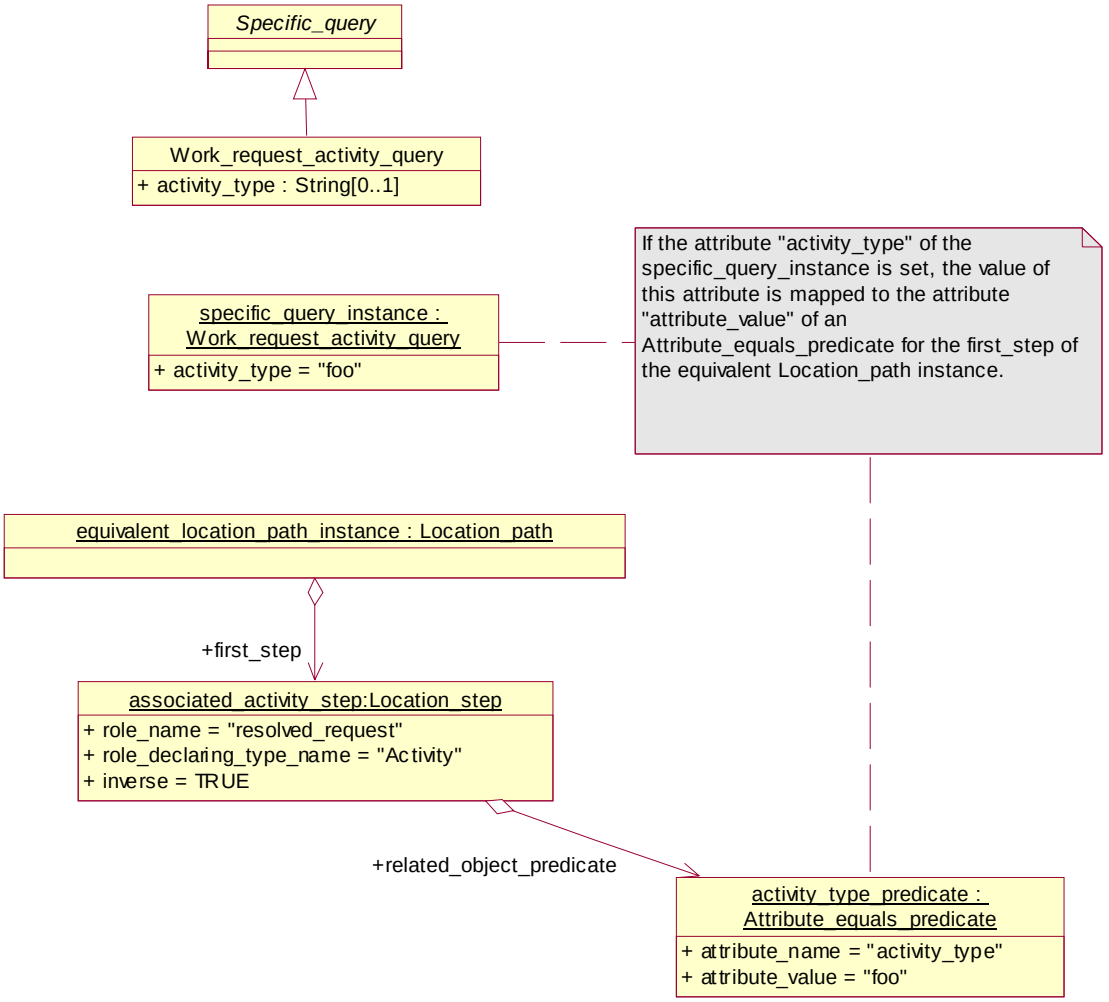


**Figure 9.84 - Definition, sample instance and equivalent Location_path instance of the Work_request_activity_query**

## 9.7.69 Work_request_query

The Work_request_query selects Work_request objects.

### *Parameters*

- id : String [0..1]
- request_type : String [0..1]
- status : String [0..1]
- version_id : String [0..1]
- classification_role : String [0..1]
- classification_id : String [0..1]

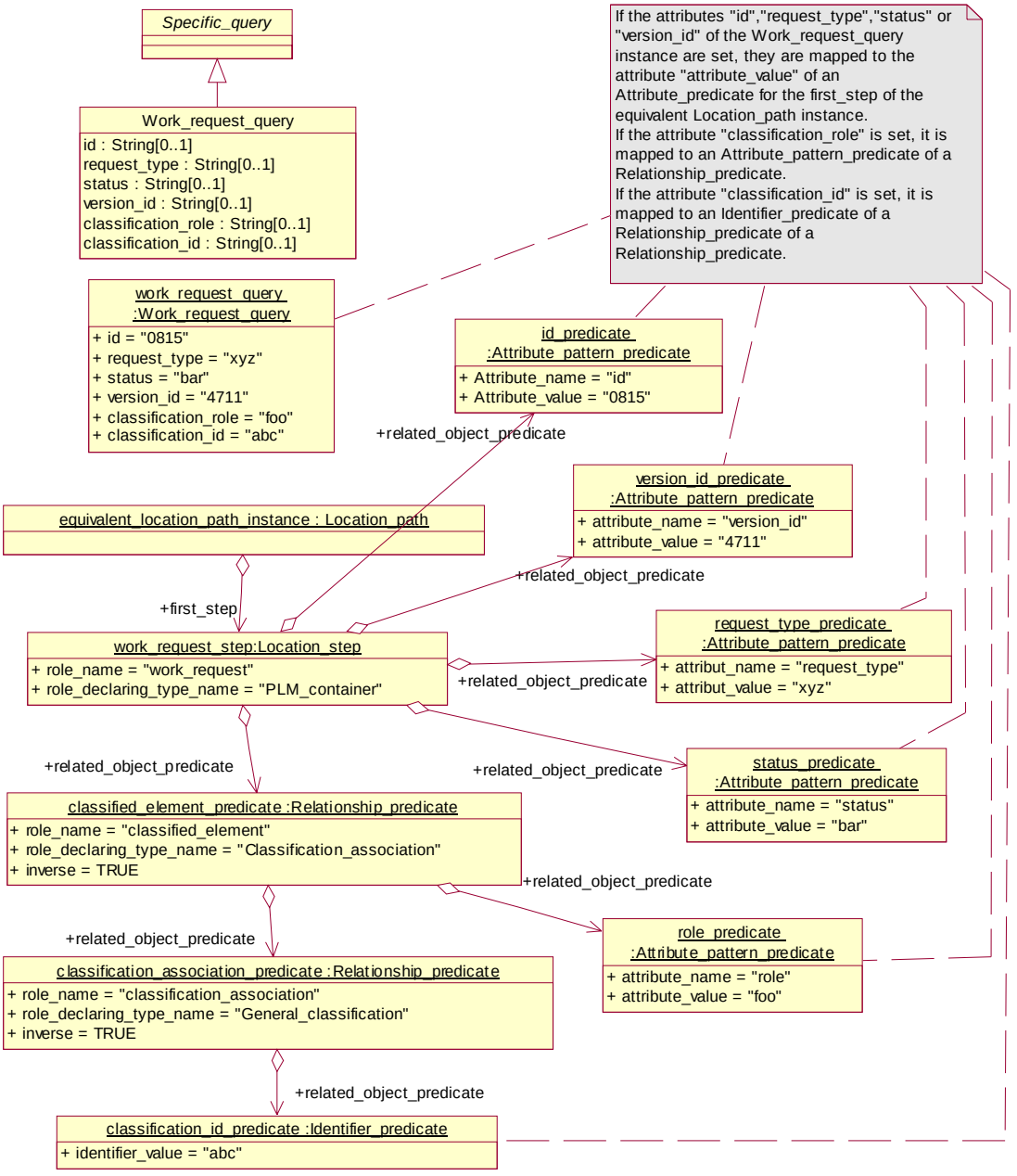


**Figure 9.85 - Definition, sample instance and equivalent Location_path instance of the Work_request_query**

### 9.7.70 Work_request_relationship_query

The Work_request_relationship_query traverses from Work_request objects via Work_request_relationship objects to Work_request objects.

***Parameters***

- relation_type : String [0..1]
- inverse : Boolean [0..1]

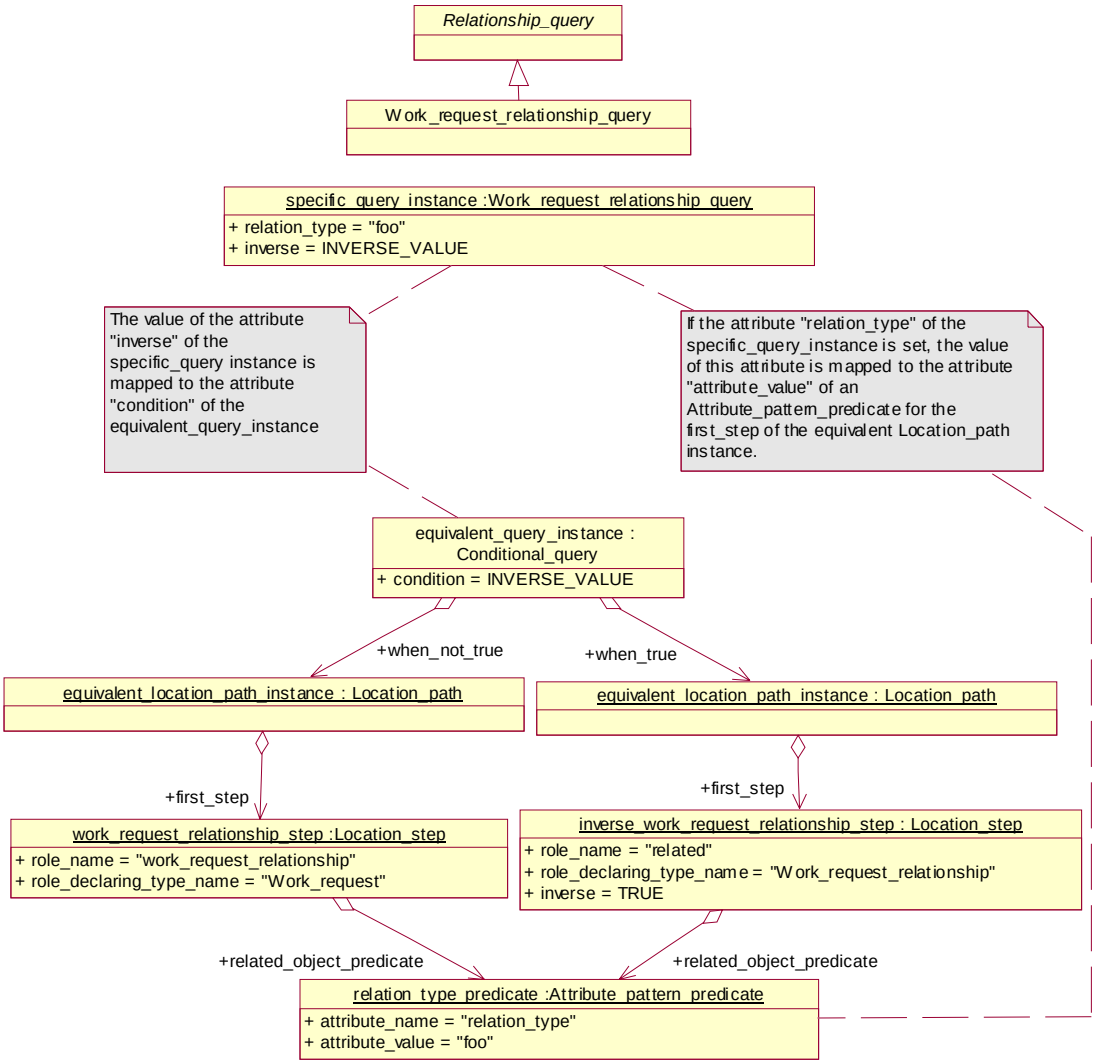


**Figure 9.86 - Definition, sample instance and equivalent Location_path instance of the Work_request_relationship_query**

### 9.7.71 Work_request_scope_query

The Work_request_scope_query traverses from Work_request objects to the Activity_element_select objects which are the scope of the Work_request objects.

***Parameters***

- none

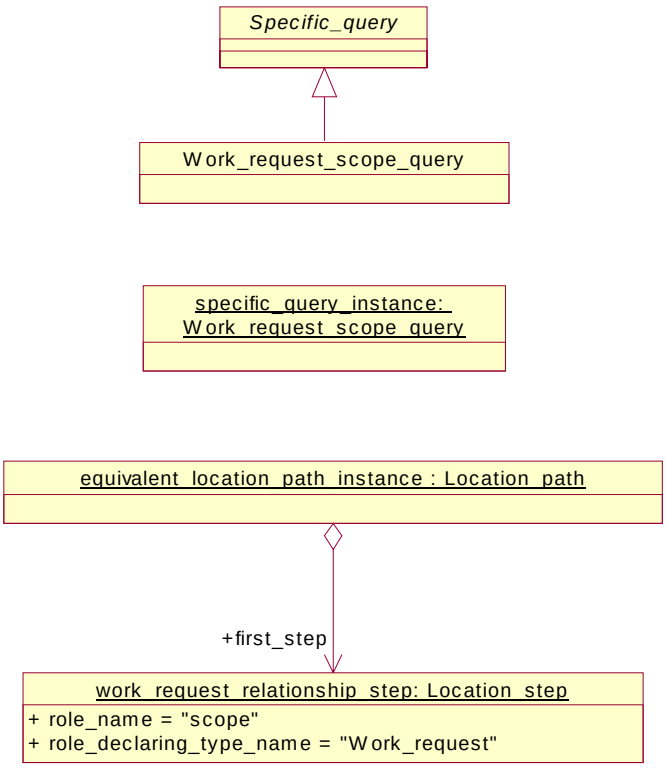


**Figure 9.87 - Definition, sample instance and equivalent Location_path instance of the Work_request_scope_query**

## 9.8 PDTnet Queries Conformance Point

The PDTnet conformance point defines a set of specialized queries that fulfill the requirements of the use cases described in Section 7.2 . The semantic of each specialized query of this conformance point is defined by an equivalent Location_path instance. The semantic of Location_path is defined in the Generic Queries Conformance Point, Chapter 9.3.

The semantics of each specialized query of this conformance point is defined by an equivalent Query instance from the Specific Queries Conformance Point. All queries of the PDTnet Queries Conformance Point extends directly or indirectly the abstract base class Pdtnet_query.

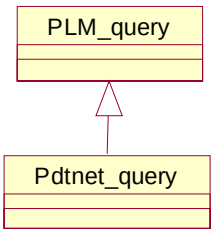


**Figure 9.88 - Class diagram of the Pdtnet_query**

### 9.8.1 General_detail_query

The General_detail_query returns general detail information from objects selected by a uid.

Parameters:

- uid : UID
- relating_type_name : String [0..*]
- add_aliases : Boolean [0..1]
- lias_id : String [0..1]
- add_authorizations : Boolean [0..1]
- authorization_role : String [0..1]
- add_dates : Boolean [0..1]
- date_role : String [0..1]
- add_properties : Boolean [0..1]
- property_name : String [0..1]
- add_classifications : Boolean [0..1]
- classification_role : String [0..1]
- add_approvals : Boolean [0..1]
- approval_level : String [0..1]
- add_activities : Boolean [0..1]
- activity_role : String [0..1]
- add_effectivities : Boolean [0..1]
- effectivity_role : String [0..1]

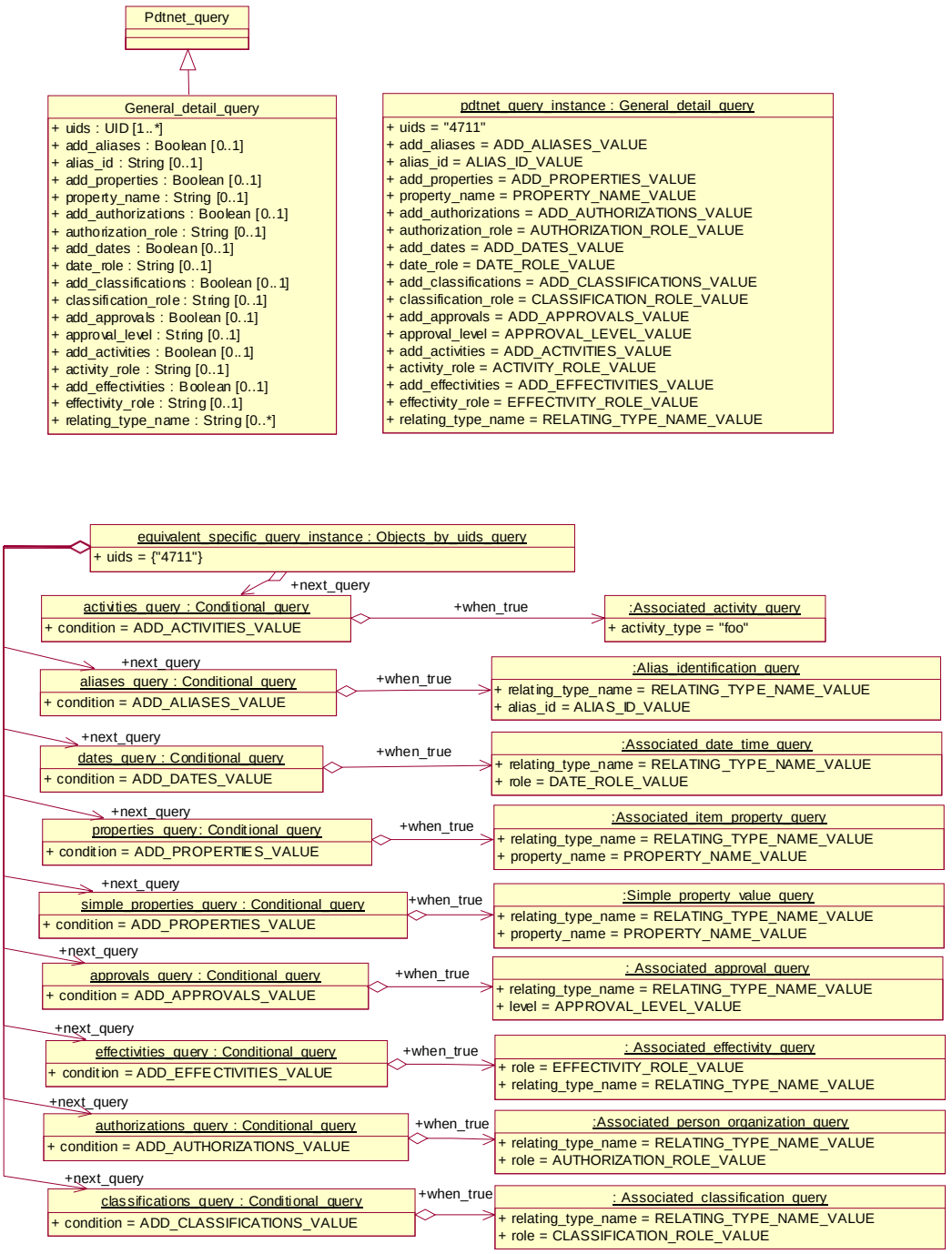


**Figure 9.89 - Definition, sample instance and equivalent specific query instance of the General_detail_query**

### 9.8.2 Document_detail_query

The Document_detail_query returns detail information of an Document, Document_version, or Document_representation object selected by a uid.

#### *Parameters (without inherited)*

- classification_name : String [0..1]
- add_versions : Boolean [0..1]
- version_id : String [0..1]
- add_representations : Boolean [0..1]

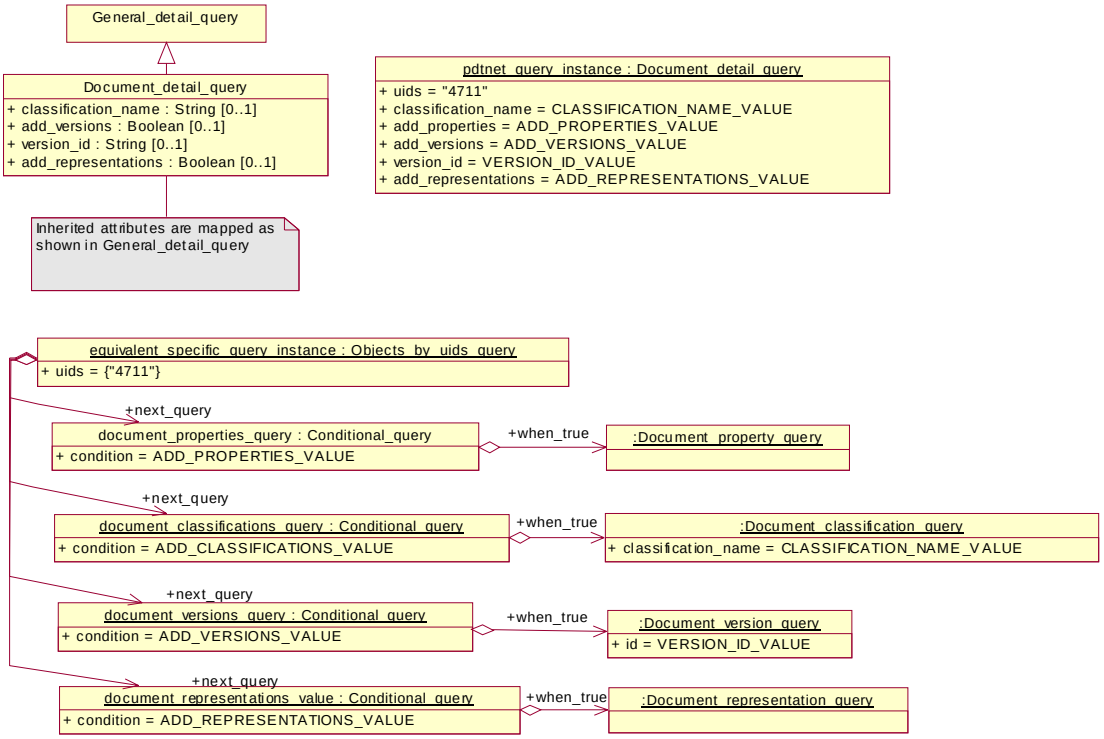


**Figure 9.90 - Definition, sample instance and equivalent specific query instance of the Document_detail_query**

### 9.8.3 Document_selection_query

The Document_selection_query selects objects of class Document and includes related Document_version and Document_representation objects.

#### *Parameters*

- name : String [0..1]

- id : String [0..1]
- version_id : String [0..1]
- classification_name : String [0..1]

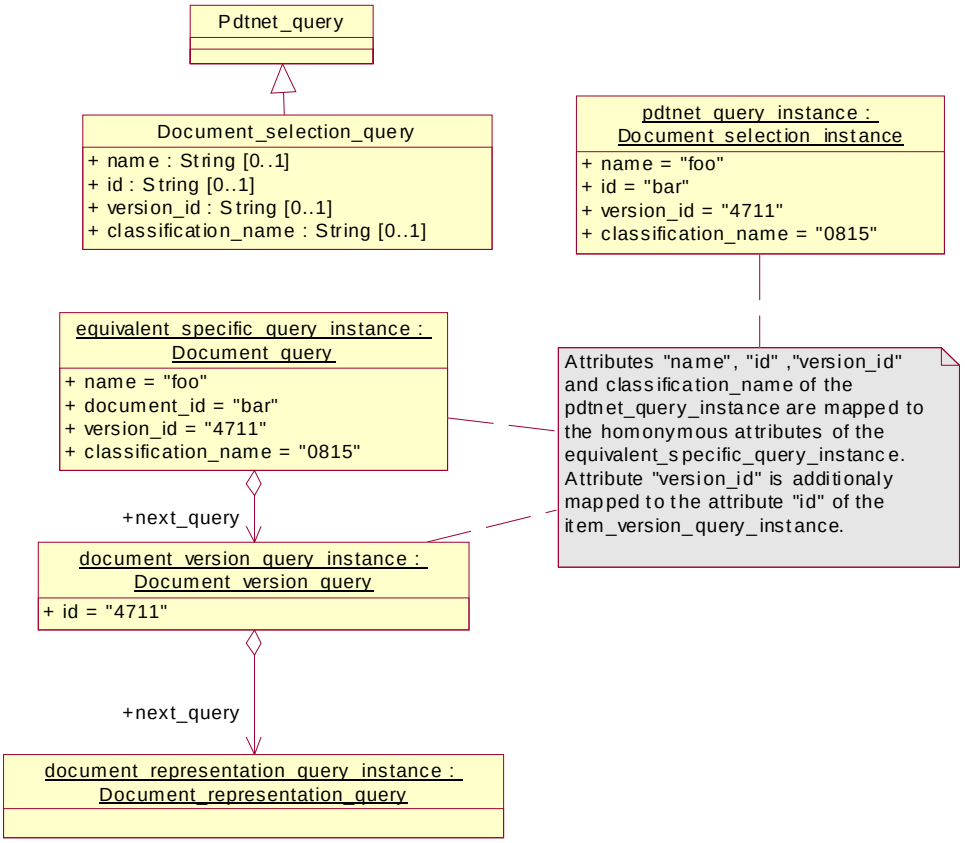


**Figure 9.91 - Definition, sample instance and equivalent specific query instance of the Document_selection_query**

### 9.8.4 Document_ traversal_query

The Document_traversal_query traverses from a Document_representation object selected by its uid via Document_structure objects to related Document_representation objects in a document structure.

#### *Parameters*

- uid : UID
- relation_type : String [0..1]

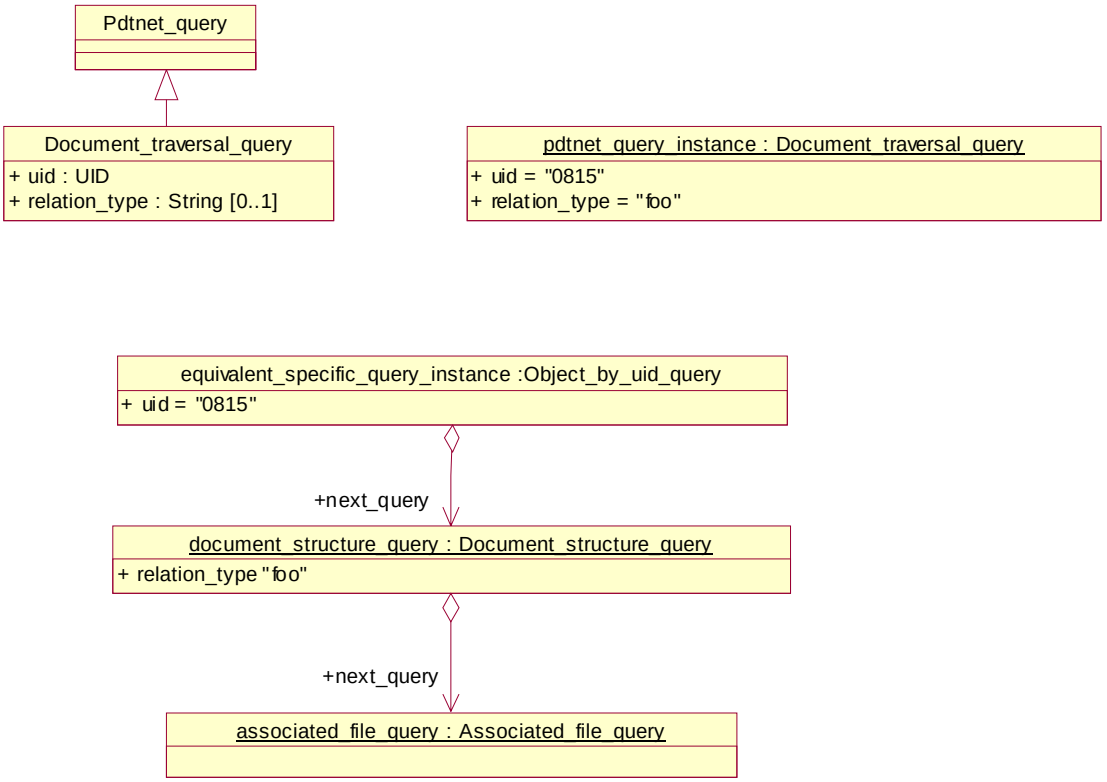


**Figure 9.92 - Definition, sample instance and equivalent specific query instance of the Document_traversal_query**

### 9.8.5 Item_detail_query

The Item_detail_query returns detail information of an Item, Item_version, Design_discipline_item_definition, Item_instance or Assembly_component_relationship object selected by a uid.

#### *Parameters (without inherited)*

- add_version_relationships : Boolean [0..1]
- version_relationship_type : String [0..1]
- add_documents : Boolean [0..1]
- document_role : String [0..1]
- classification_name : String [0..1]
- add_placement : Boolean [0..1]

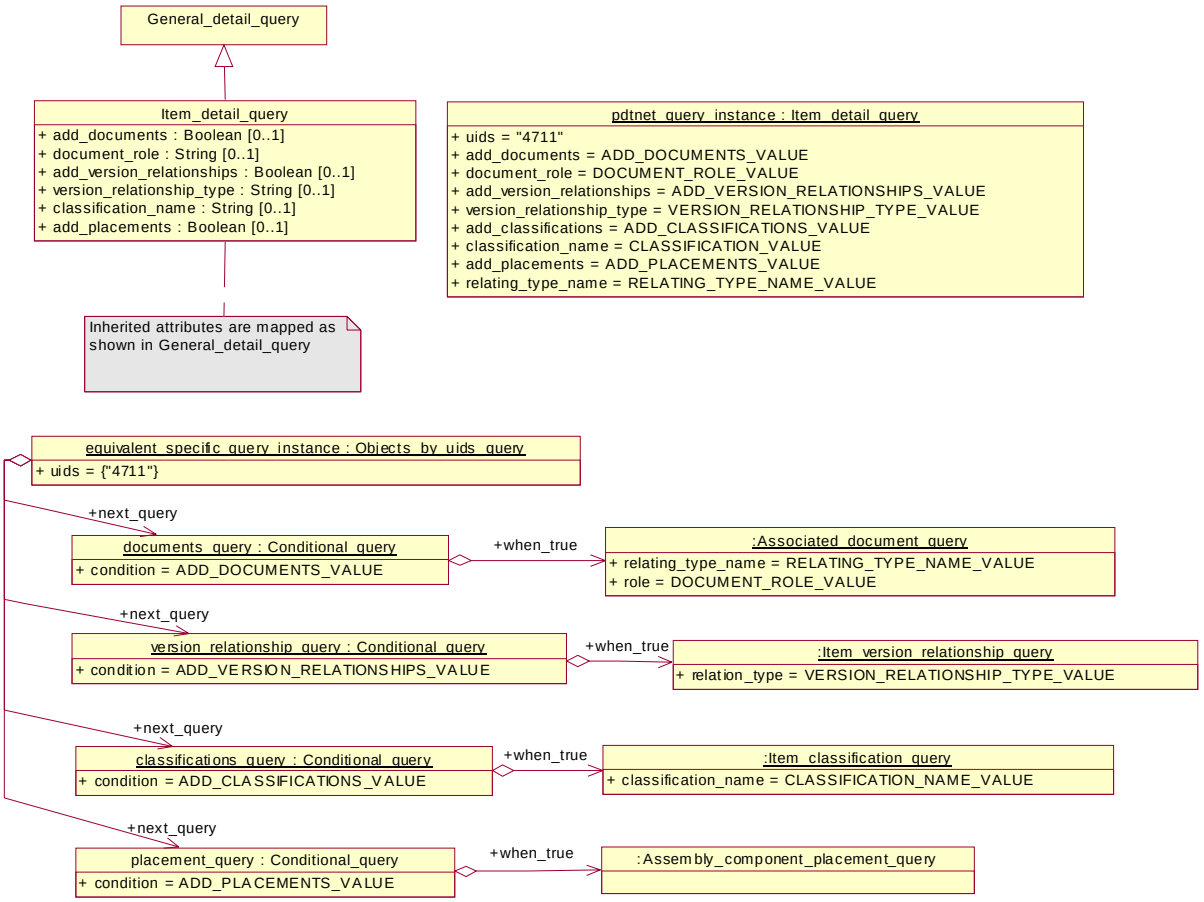


**Figure 9.93 - Definition, sample instance and equivalent specific query instance of the Item_detail_query**

## 9.8.6 Item_selection_query

The Item_selection_query selects objects of class Item and includes related Item_version and Design_discipline_item_definition objects.

### *Parameters*

- name : String [0..1]
- id : String [0..1]
- version_id : String [0..1]
- classification_name : String [0..1]

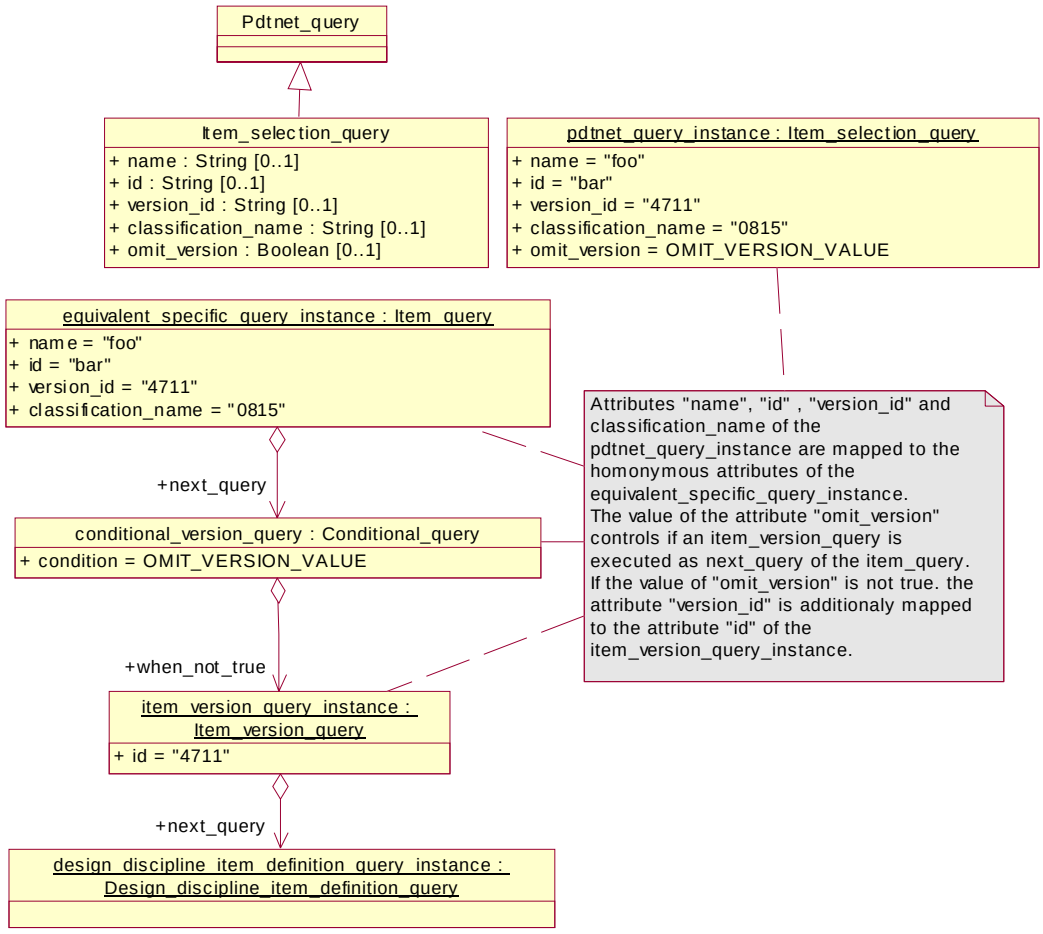


**Figure 9.94 - Definition, sample instance and equivalent specific query instance of the Item_selection_query**

## 9.8.7 Item_ traversal_query

The Item_traversal_query traverses from a Design_discipline_item_definition object to the higher or lower Design_discipline_item_definition objects in an assembly structure.

### *Parameters*

- uid : UID
- inverse_direction : Boolean [0..1]

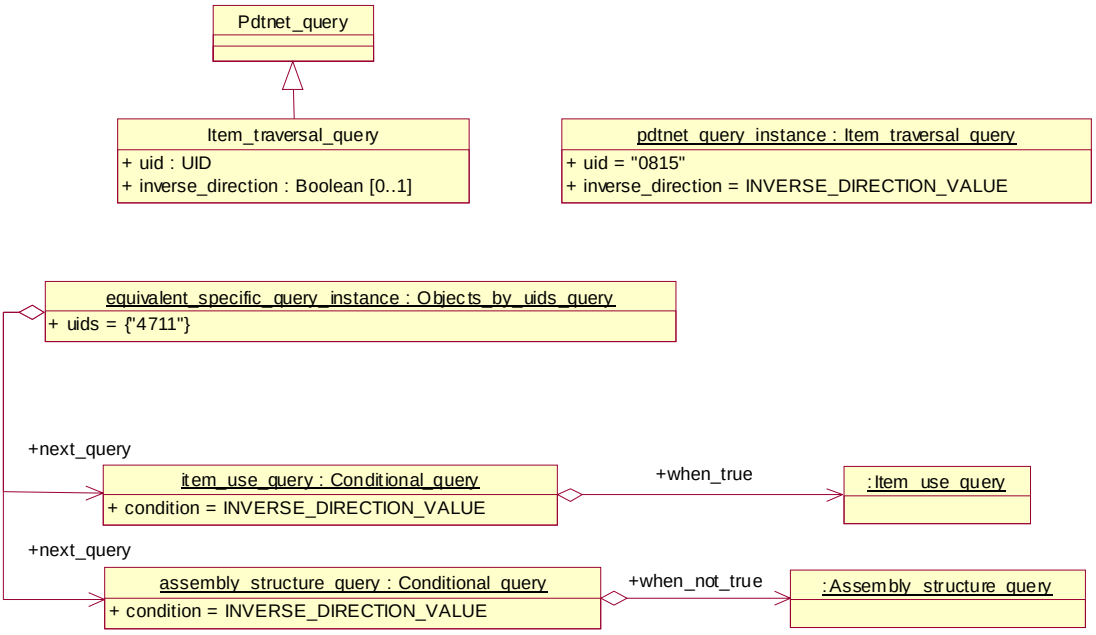


**Figure 9.95 - Definition, sample instance and equivalent specific query instance of the Item_traversal_query**

### 9.8.8 Product_ detail_query

The Product_detail_query returns detail information of an Complex_product object selected by a uid.

***Parameters (without inherited)***

- add_configurations : Boolean [0..1]
- configuration_type : String [0..1]
- add_documents : Boolean [0..1]
- document_role : String [0..1]

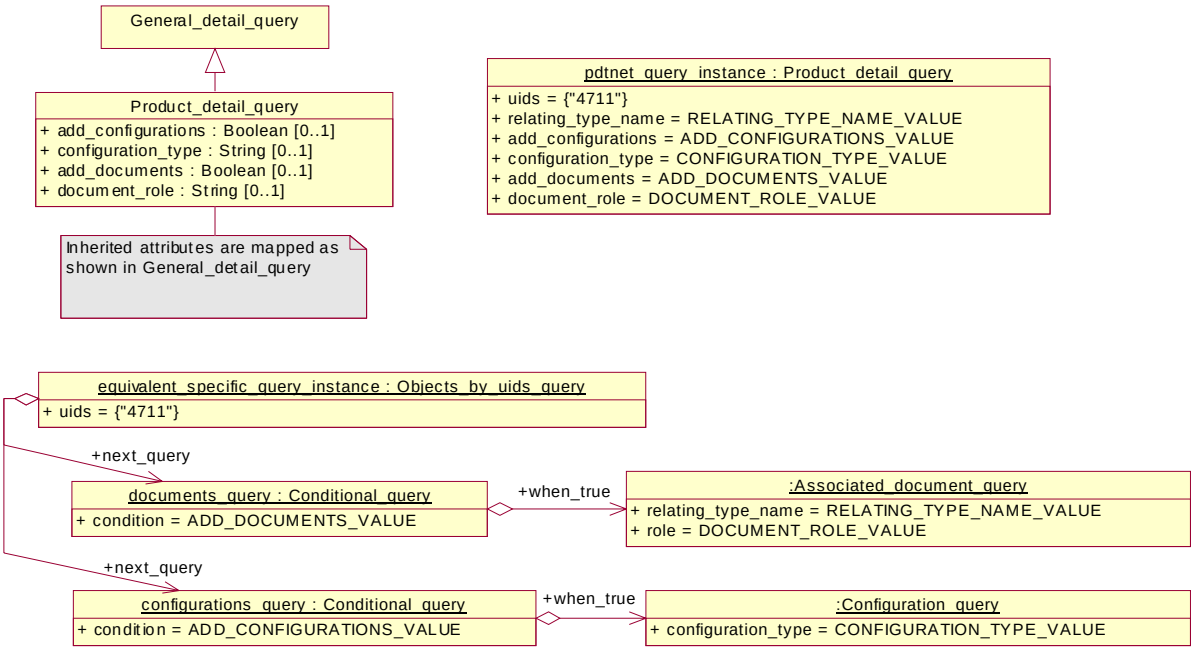


**Figure 9.96 - Definition, sample instance and equivalent specific query instance of the Product_detail_query**

### 9.8.9 Product_selection_query

The Product_selection_query selects objects of class Product_class and includes Product_function_component_select related via Class_structure_relationship objects.

#### *Parameters*

- name : String [0..1]
- id : String [0..1]

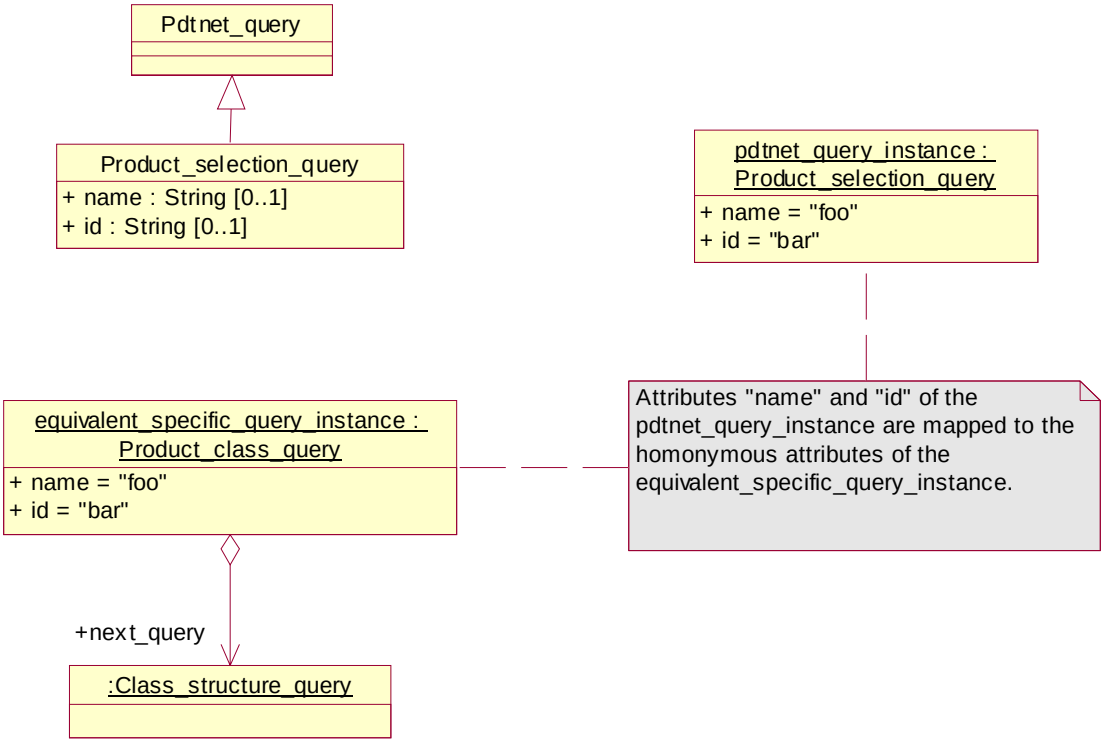


**Figure 9.97 - Definition, sample instance and equivalent specific query instance of the Product_selection_query**

### 9.8.10 Product_traversal_query

The Product_traversal_query traverses from a Complex_product object selected by its uid via Product_structure_relationship or Alternative_solution_objects to related Complex_product or Item_instance objects in a product structure.

#### *Parameters*

- uid : UID
- relation_type : String [0..1]

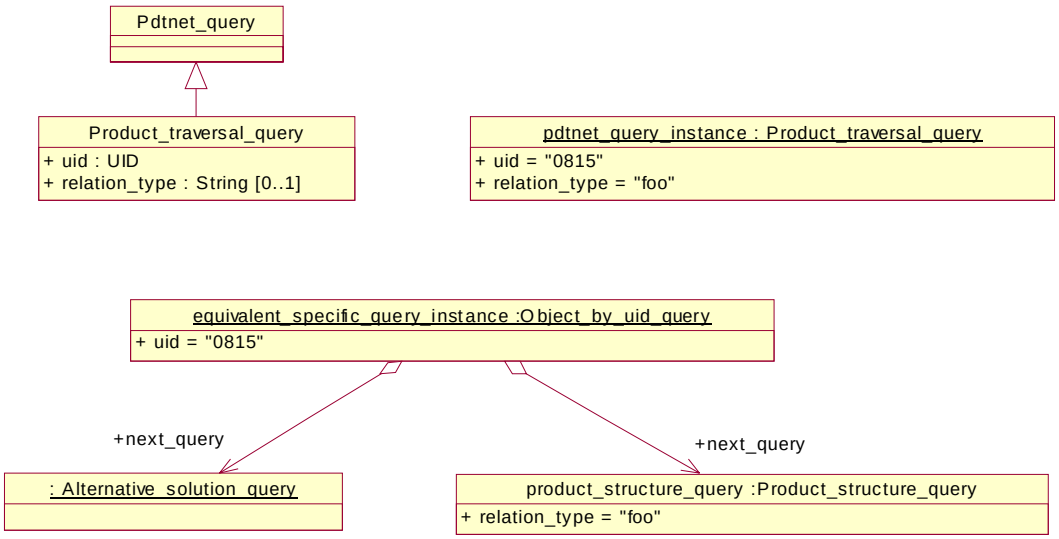


**Figure 9.98 - Definition, sample instance and equivalent specific query instance of the Product_traversal_query**

### 9.8.11 Work_order_selection_query

The Work_order_selection_query selects objects of class Work_order.

#### *Parameters*

- Id: String [0..1]
- work_order_type: String [0..1]
- version_id: String [0..1]
- classification_role: String [0..1]
- classification_id: String [0..1]
- description: String [0..1]

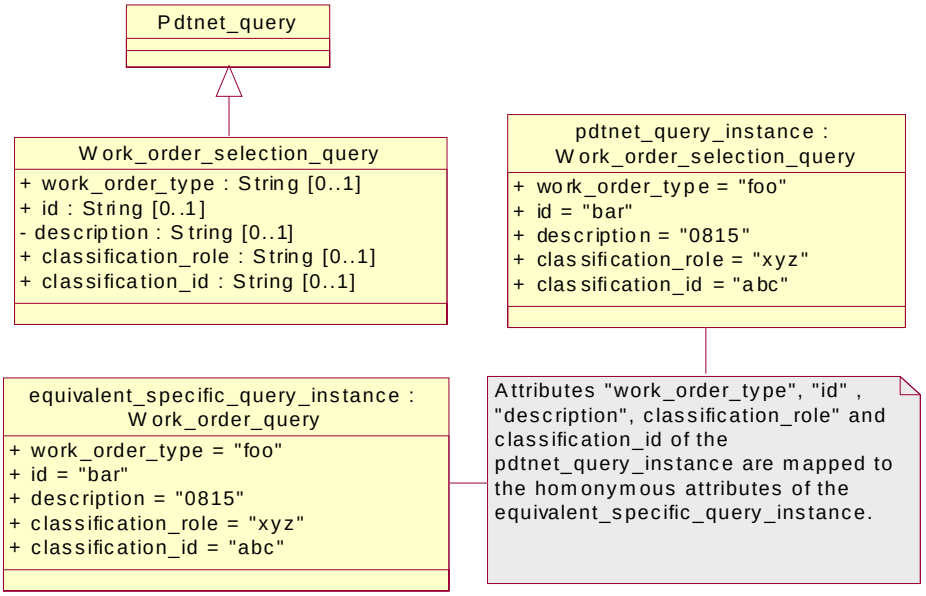


**Figure 9.99 - Definition, sample instance and equivalent specific query instance of the Work_order_selection_query**

### 9.8.12 Work_order_detail_query

The Work_order_detail_query returns detail information of a Work:order object selected by a uid.

#### *Parameters (without inherited)*

- add_documents : Boolean [0..1]
- document_role : String [0..1]

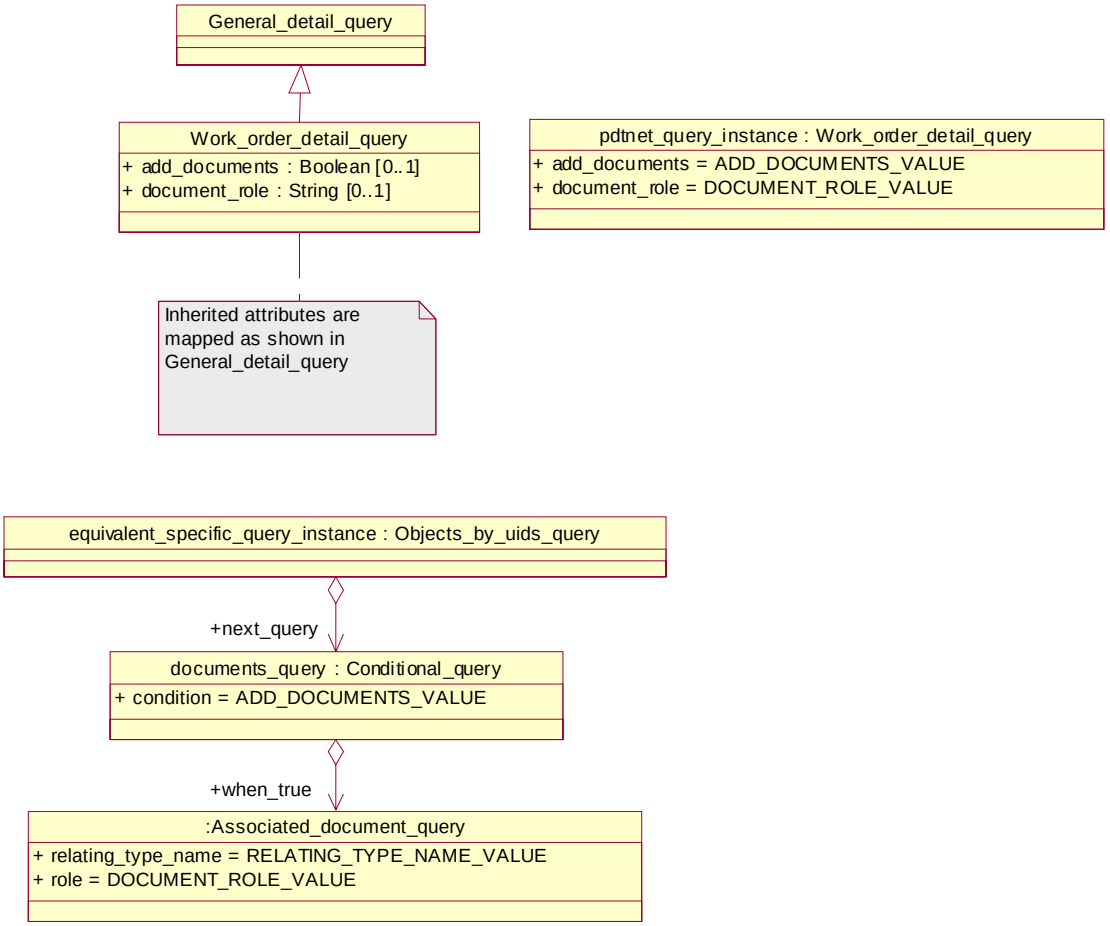


**Figure 9.100 - Definition, sample instance and equivalent specific query instance of the Work_order_detail_query**

### 9.8.13 Work_request_selection_query

The Work_order_selection_query selects objects of class Work_order.

***Parameters***

- Id : String [0..1]
- request: String [0..1]
- version_id: String [0..1]
- status: String [0..1]

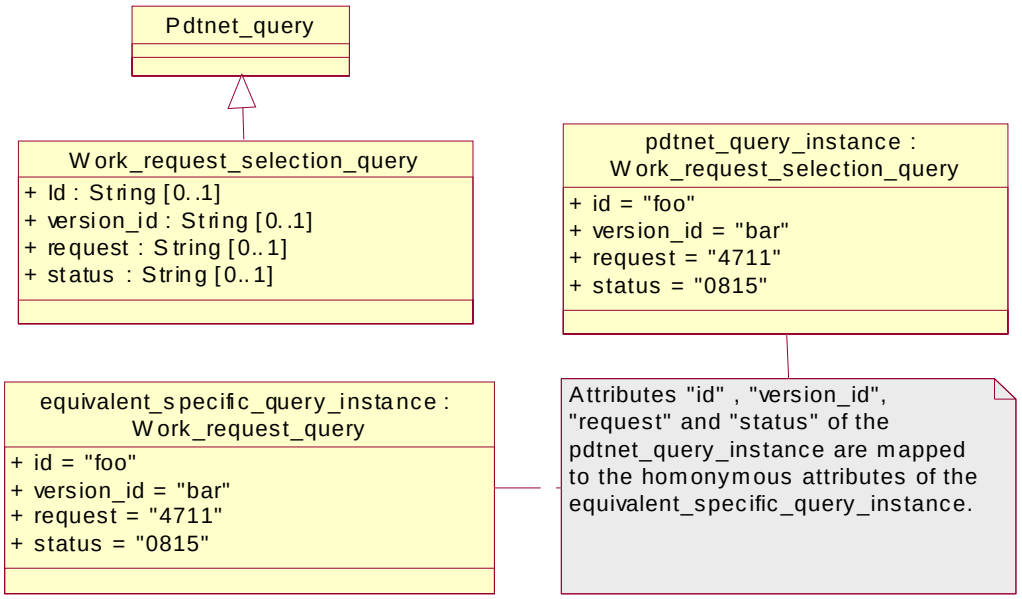


**Figure 9.101 - Definition, sample instance and equivalent specific query instance of the Work_request_selection_query**

### 9.8.14 Work_request_traversal_query

The Work_request_traversal_query traverses from Work_request objects via Work_request_relationship objects to other Work_request objects.

Parameters:

- uid : UID
- relation_type: String [0..1]

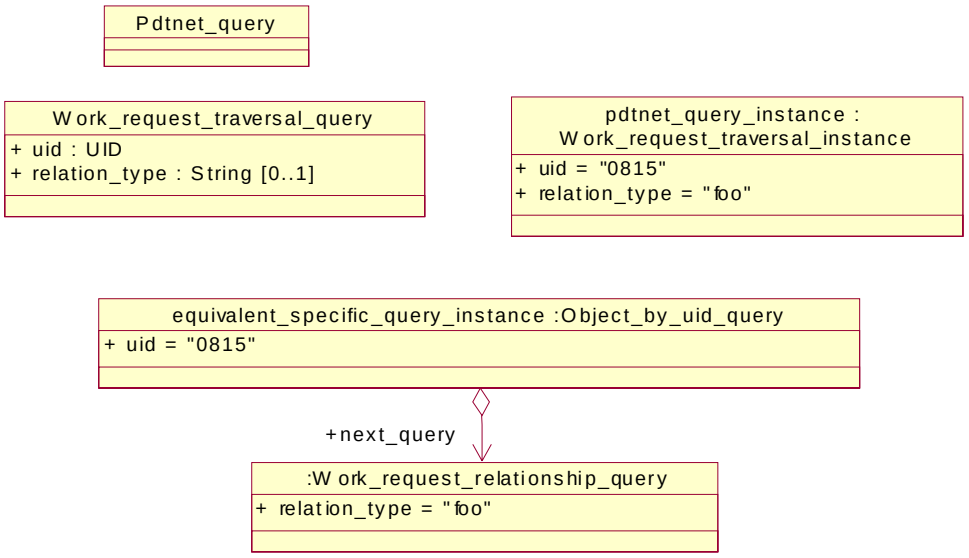


**Figure 9.102 - Definition, sample instance and equivalent specific query instance of the Work_request_traversal_query**

### 9.8.15 Work_request_detail_query

The Work_order_detail_query returns detail information of a Work:order object selected by a uid.

#### *Parameters (without inherited)*

- add_documents : Boolean [0..1]
- document_role : String [0..1]

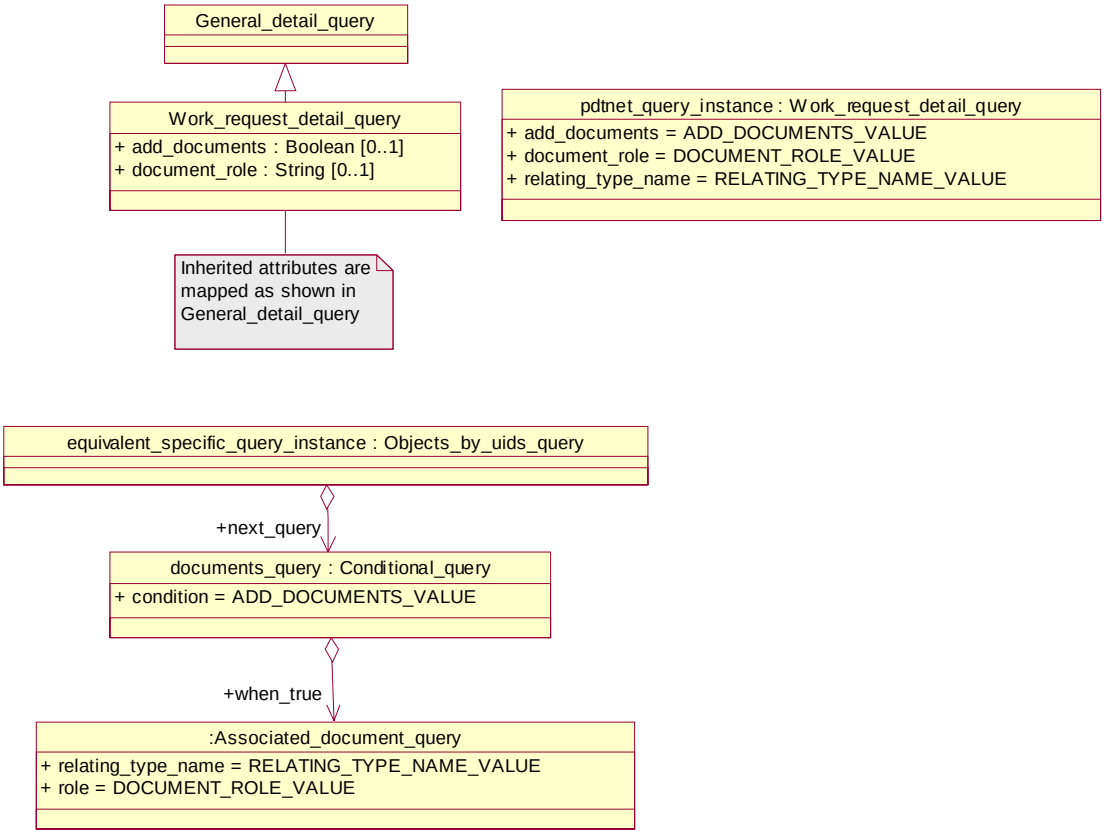


**Figure 9.103 - Definition, sample instance and equivalent specific query instance of the Work_request_detail_query**

## 9.9 Message Queries Conformance Point

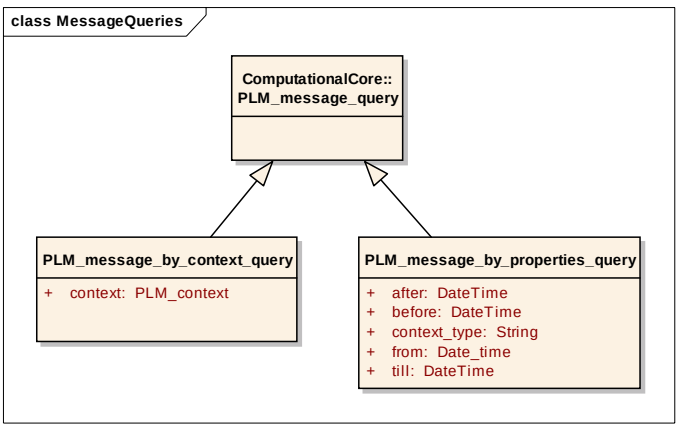


**Figure 9.104 - Message Queries**

The Message Queries Conformance Point defines queries derived from the abstract class PLM_message_query that can be used as parameters for the query_messages operation of the PLM_message_connection interface.

### 9.9.1 Class Message_by_context_query

The class Message_by_context_query can be used to identifiy PLM_message objects unambiguously by their contexts.

#### *Base*

- PLM_message_query

#### *Parameters*

- PLM_context: PLM_context [0..*]

### 9.9.2 Class Message_by_properties_query

The class Message_by_properties_query can be used to query PLM_message objects by their properties.

#### *Base*

- PLM_message_query

#### *Parameters*

- omit_container : Boolean [0..1]
  if the omit_container is TRUE in all PLM_message elements of the result list of the query_messages operation the contained PLM_core_container is a NIL object.
- context_type: Type [0..1] selects messages if their contained context is an instance of the given type.
- before: dateTime [0..1] selects messages if their server incoming time is less than the given dateTime value.
- till: dateTime [0..1] selects messages if their server incoming time is less than or equals the given dateTime value.
- from: dateTime [0..1] selects messages if their server incoming time is greater than or equals the given dateTime value .
- after: dateTime [0..1] selects messages if their server incoming time is greater than the given dateTime value.

# 10 WebServices PSM

## 10.1 Overview (informative)

In the following sections a projection of the PIM into the platform specific model (PSM) with an execution infrastructure given by XML is defined. The projection is done via an enrichment of the model by a customized UML profile for XML Schema. This UML profile is given here for informal purposes.

## 10.2 UML Profile for XML Schema (informative)

To enrich the UML Informational PIM for XML Schema representation a UML profile is used. A UML profile has three key items namely stereotypes, tagged values called properties and constraints.

### 10.2.1 UML Model

On the entire UML model level the stereotype << XSDschema >> is applied. It can have the following tagged values:

**Table 10.26 - Stereotype <<XSDschema>>**

| Can be applied to | UML model | |
|---|---|---|
| **Property** | **Value** | **Description** |
| targetNamespace | namespace URI | The URI which uniquely identifies this schema's namespace. |
| elementFormDefault | qualified<br>unqualified | Specifies whether elements are quali-fied or unqualified. |
| attributeFormDefault | qualified<br>unqualified | Specifies whether attributes are quali-fied or unqualified. |
| version | string value | The version of this schema. |
| modelGroup | all<br>sequence<br>choice<br>none<br>omitComplexType | Specifies the content model used for generating complexType definitions. |
| globalElement | true<br>false | Specifies if global element declarations are created for complex types. |
| attributeMapping | element<br>attribute | Specifies the mapping for UML attrib-utes. |
| roleMapping | element<br>attribute | Specifies the mapping for roles of UML associations. |
| anonymousRole | true<br>false | Specifies if role names of UML attrib-utes are mapped to elements. |
| anonymousType | true<br>false | Specifies if the types of UML attributes are mapped to elements. |

**Table 10.26 - Stereotype <<XSDschema>>**

| | | |
|---|---|---|
| typeContainment | true<br>false | Specifies if types are contained instead of referencing them. |
| elementNamingMapping | firstLetterUpperCase<br>firstLetterLowerCase<br>upperCamelCase<br>lowerCamelCase<br>omitElement | Specifies the naming for elements. |
| attributeNamingMapping | firstLetterUpperCase<br>firstLetterLowerCase<br>upperCamelCase<br>lowerCamelCase<br>omitAttribute | Specifies the naming for attributes. |

The four stereotypes XSDschema, XSDtranslatableString, XSDelement, and XSDattribute form a hierarchy in that order. The derived stereotypes need to redefine only those values of the named properties which require new values.

**Example**

```
Model "PLM_services"
Stereotype << XSDschema >>
targetNamespace = urn:omg.org/plm20/informational/core
elementFormDefault      = qualified
attributeFormDefault    = unqualified
version                 = 2.0
modelGroup              = sequence
globalElement           = false
attributeMapping        = element
roleMapping             = element
anonymousRole           = false
anonymousType           = false
typeContainment         = false
elementNamingMapping    = firstLetterUpperCase
attributeNamingMapping  = firstLetterLowerCase
```

```
<xs:schema
    targetNamespace='urn:omg.org/plm20/informational/core'
    xmlns:ns2='urn:omg.org/plm20/informational/core'
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="2.0">
```

### 10.2.2 UML Package

On the UML package “Multi_language_support” the stereotype << XSDtranslatableString >> is applied. It doesn't have any tagged values.

XML provides its own mechanism to specify the language used in the contents and attribute values of any element in a XML document, the predefined attribute xml:lang. Based on this a XML specific concept has been developed to map the multi language support for string values of the PIM model.

If the UML Interface “String_select” is used by any UML composition, the type “Translatable_string” is used in XML instead. Therefore the predefined complex types “Translatable_string,” “Translation,” and “Translations” are introduced in the XML schema.

**Table 10.27 - Stereotype << XSDtranslatableString>>**

| | |
|---|---|
| Can be applied to | UML package |

**Example**

```
Package "Multi_language_support"
Stereotype << XSDtranslatableString >>
```

```
<xs:complexType name="Item">
    …
    <xs:element name="Name" type="Translatable_string"/>
    …
</xs:complexType>

<xs:complexType name="Translatable_string">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="translations" type="xs:IDREF" use="optional"/>
                <xs:annotation>
                    <xs:documentation>REFERENCE TO Translations</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute ref="xml:lang" use="optional"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:complexType name="Translation">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute ref="xml:lang" use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:complexType name="Translations">
    <xs:complexContent>
        <xs:extension base="PLM_root_object">
            <xs:sequence>
                <xs:element name="Translation" type="Translation" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The mapping of instance values from UML to XML is as follows:

**Table 10.28 - Mapping of instance values from UML to XML**

| UML | XML |
|---|---|
| Default_language_string | Translatable_string |
| Multi_language_string<br>.primary_language_dependent_string<br>.String_with_language.contents | Translatable_string |
| Multi_language_string<br>.primary_language_dependent_string<br>.String_with_language<br>.language_specification<br>.Language.language_code | Translatable_string /@xml:lang |
| Multi_language_string<br>.primary_language_dependent_string<br>.String_with_language<br>.language_specification<br>.Language.country_code | Translatable_string /@xml:lang |
| Multi_language_string<br>.additional_language_dependent_string<br>.String_with_language.contents | Translation |
| Multi_language_string<br>.additional_language_dependent_string<br>.String_with_language<br>.language_specification<br>.Language.language_code | Translation /@xml:lang |
| Multi_language_string<br>.additional_language_dependent_string<br>.String_with_language<br>.language_specification<br>.Language.country_code | Translation /@xml:lang |

### 10.2.3 UML Classes

On each UML class the stereotype << XSDcomplexType >> is applied. It can have the following tagged values:

**Table 10.29 - Stereotype <<XSDcomplexType>>**

| Can be applied to | UML class | |
|---|---|---|
| **Property** | **Value** | **Description** |
| modelGroup | all<br>sequence<br>choice<br>multiChoice<br>omitComplexType | Specifies the content model used for generating this complexType definition. |

**Table 10.29 - Stereotype <<XSDcomplexType>>**

| | | |
|---|---|---|
| globalElement | true<br>false | Specifies if a global element declaration is created for this complexType. |
| attributeMapping | element<br>attribute | Specifies the mapping for UML attributes within this complexType. |
| roleMapping | element<br>attribute | Specifies the mapping for roles of UML associations within this complexType. |
| anonymousRole | true<br>false | Specifies if the role names of UML attributes are mapped to elements within this complex type. |
| anonymousType | true<br>false | Specifies if the types of UML attributes are mapped to elements within this complex type. |
| typeContainment | true<br>false | Specifies if types are contained instead of referencing them within this complex type. |
| elementNamingMapping | firstLetterUpperCase<br>firstLetterLowerCase<br>upperCamelCase<br>lowerCamelCase<br>omitElement | Specifies the naming for elements within this complex type. |
| attributeNamingMapping | firstLetterUpperCase<br>firstLetterLowerCase<br>upperCamelCase<br>lowerCamelCase<br>omitAttribute | Specifies the naming for attributes within this complex type. |

Each of the above named properties shall apply to all UML classes, attributes, associations and compositions which do not have an own stereotype overwriting these values.

### Generalization

Only single inheritance is treated by the UML to XML Schema mapping. This is sufficient since the PIM UML model does not contain any multiple inheritance. Each subclass will be a complexType with complexContent and extension base=”superclass”. Abstract classes are mapped to complex types which are abstract.

### Example

```
Class "PLM_container"
Stereotype << XSDcomplexType >>
modelGroup              = multiChoice
globalElement           = true
```

```
<xs:element name="PLM_container" type="PLM_container"/>
<xs:complexType name="PLM_container">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        …
    </xs:choice>
</xs:complexType>
```

### 10.2.4 UML Interfaces

UML interfaces are not treated by the UML to XML Schema mapping since the interfaces are only referenced by other classes. These references are mapped to XML schema references of type IDREF and IDREFS, which point to the underlying types of an interface.

### 10.2.5 UML Attributes, Associations and Compositions

On each UML attribute, association and composition the stereotypes << XSDelement >> or << XSDattribute >> are applied. They can have the following tagged values:

**Table 10.30 - Stereotype <<XSDelement>>**

| Can be applied to | UML attribute,<br>UML association,<br>UML composition | |
|---|---|---|
| **Property** | **Value** | **Description** |
| position | integer value | Causes the elements to be ordered within a sequence model group of the containing complexType. |
| anonymousRole | true<br>false | Specifies if the role name of a UML attribute is mapped to an element. |
| anonymousType | true<br>false | Specifies if the type of a UML attribute is mapped to an element. |
| typeContainment | true<br>false | Specifies if the type is contained instead of referencing it. |
| elementNamingMapping | firstLetterUpperCase<br>firstLetterLowerCase<br>upperCamelCase<br>lowerCamelCase<br>omitElement | Specifies the naming for this element. |

**Example**

```
Attribute "relation_type"
Stereotype << XSDelement >>
position                = 03
anonymousRole           = false
anonymousType           = true
typeContainment         = true

Composition "description"
Stereotype << XSDelement >>
position                = 02
anonymousRole           = false
anonymousType           = true
typeContainment         = true
```

```
Composition "change"
Stereotype << XSDelement >>
position                = 04
anonymousRole           = true
anonymousType           = false
typeContainment         = true

Association "related"
Stereotype << XSDelement >>
position                = 01
anonymousRole           = false
anonymousType           = false
typeContainment         = false
```

```
<xs:complexType name="Item_version_relationship">
    <xs:complexContent>
        <xs:extension base="PLM_object">
            <xs:sequence>
                <xs:element name="Related" type="xs:IDREF">
                    <xs:annotation>
                        <xs:documentation>REFERENCE TO Item_version</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="Description" type="Translatable_string" minOccurs="0"/>
                <xs:element name="Relation_type" type="xs:string"/>
                <xs:element name="Change" type="Change" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

**Table 10.31 - Stereotype <<XSDattribute>>**

| Can be applied to | UML attribute, UML association, UML composition | |
|---|---|---|
| **Property** | **Value** | **Description** |
| attributeType | qualified name | Specifies the type of the attribute. |
| use | prohibited<br>optional<br>required<br>fixed | Specifies the use of the attribute. |
| attributeNamingMapping | firstLetterUpperCase<br>firstLetterLowerCase<br>upperCamelCase<br>lowerCamelCase<br>omitAttribute | Specifies the naming for this attribute. |

### Example

```
Attribute "uid"
Stereotype << XSDattribute >>
attributeType           = xs:ID
use                     = required
```

```
<xs:complexType name="PLM_object" abstract="true">
      <xs:attribute name="uid" type="xs:ID" use="required"/>
</xs:complexType>
```

Most of the UML attributes, associations and compositions are mapped to elements in the XML Schema and also a position of these elements are needed if the modelGroup is a sequence. This is done by applying a position value to a UML attribute, association or composition.

If the type of the UML attribute is a data type it is mapped to a corresponding primitive data type of the XML Schema Definition:

| UML datatype | XSD primitive type |
|---|---|
| String | xs:string |
| Double | xs:double |
| Boolean | xs:boolean |
| Integer | xs:integer |
| UID | xs:ID |

The multiplicity of a UML attribute, association or composition is mapped to the corresponding multiplicity in the XML schema. For elements the values `minOccurs` and `maxOccurs` are used, for attributes the value `use`.

## 10.3 Informational viewpoint with applied stereotypes (normative)

The Informational viewpoint with applied stereotypes is given in a separate OMG document in XMI which is normative. Additional informative description is given below.

### 10.3.1 Informational Core

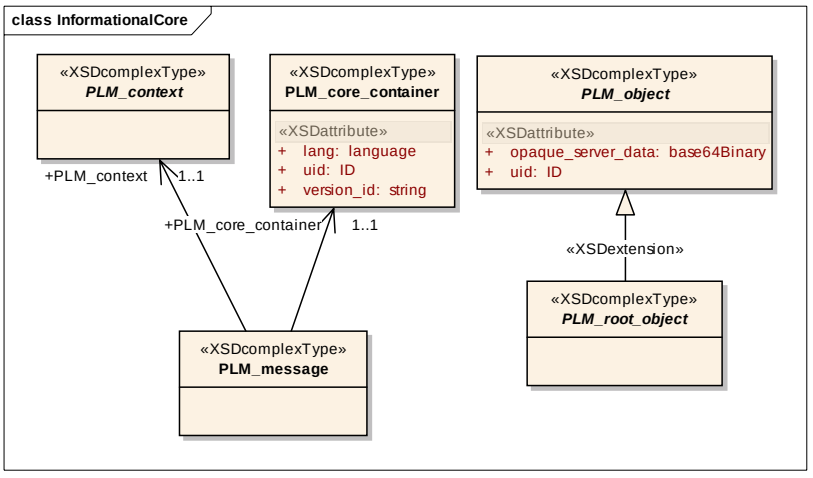


**Figure 10.1 - Informational Core**

## 10.4 Computational viewpoint with applied stereotypes (normative)

The Computational viewpoint with applied stereotypes is given in a separate OMG document in XMI which is normative. Additional informative description is given below.

### 10.4.1 Computational Core

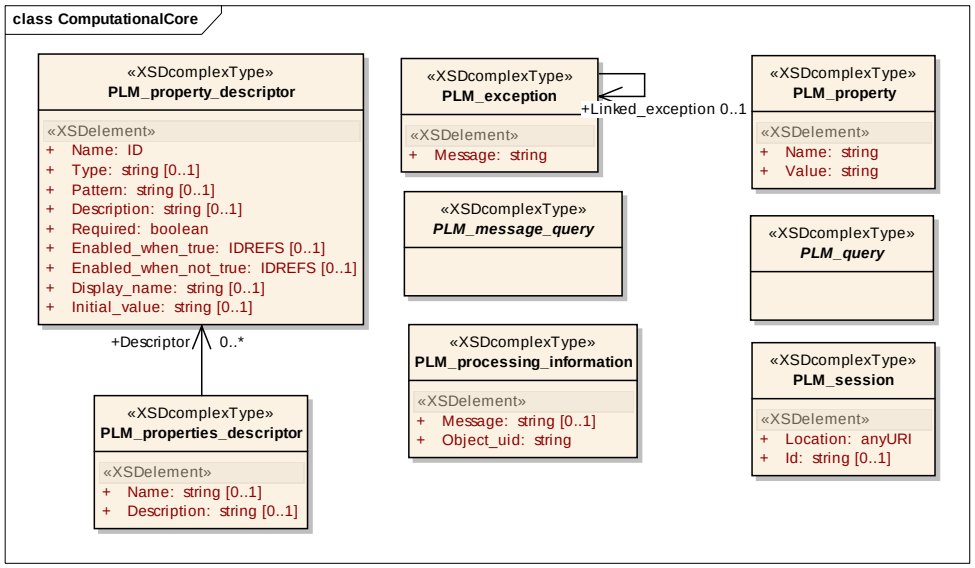


**Figure 10.2 - Computational Core**

### 10.4.2 Exceptions

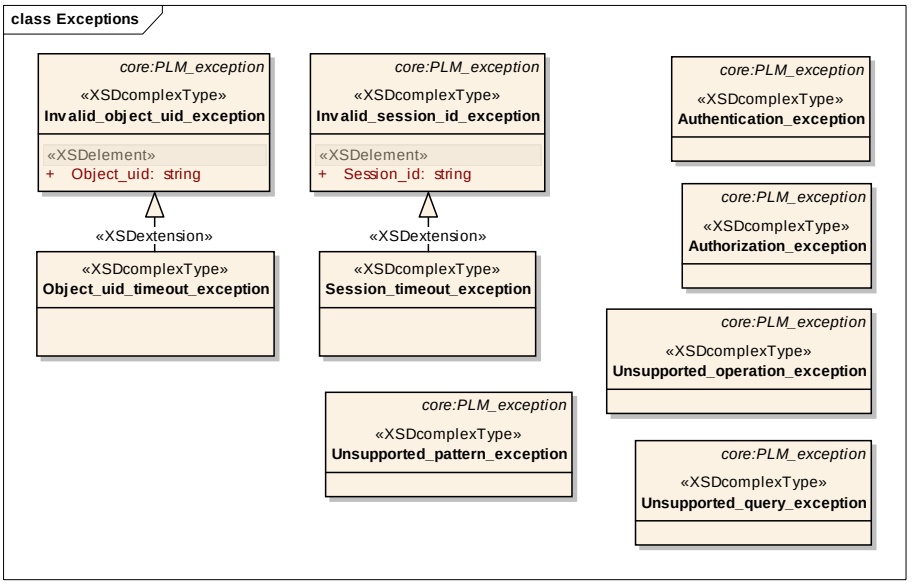


**Figure 10.3 - Exceptions**

## 10.4.3 Generic Queries

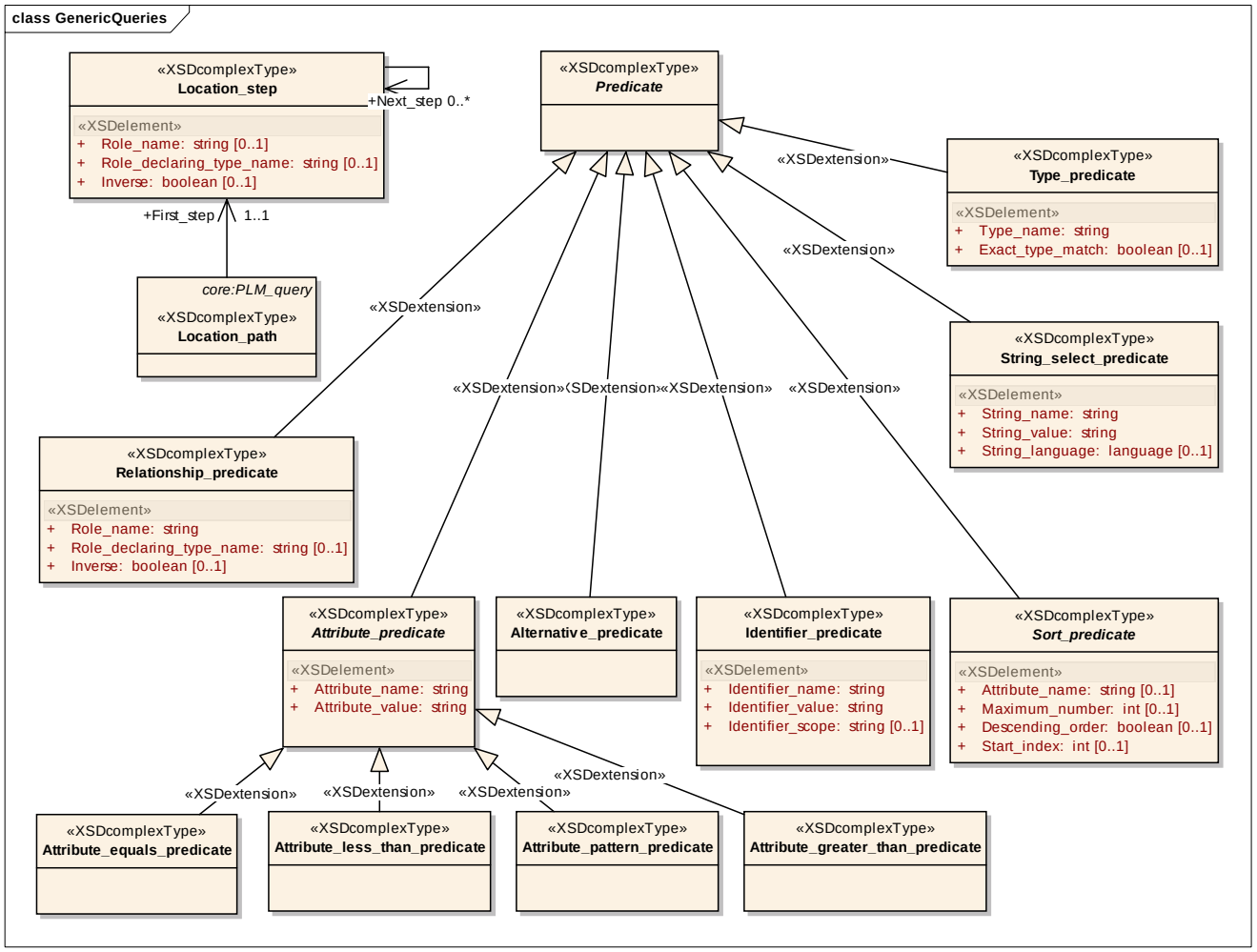


**Figure 10.4 - Generic Queries**

### 10.4.4 Informations

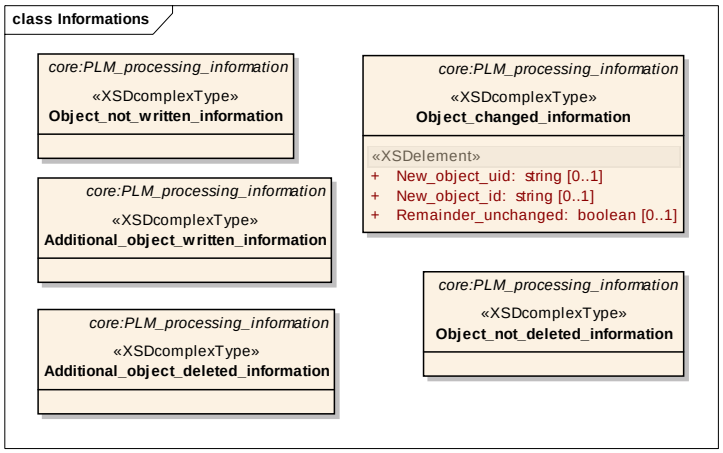


**Figure 10.5 - Informations**

### 10.4.5 Parameters

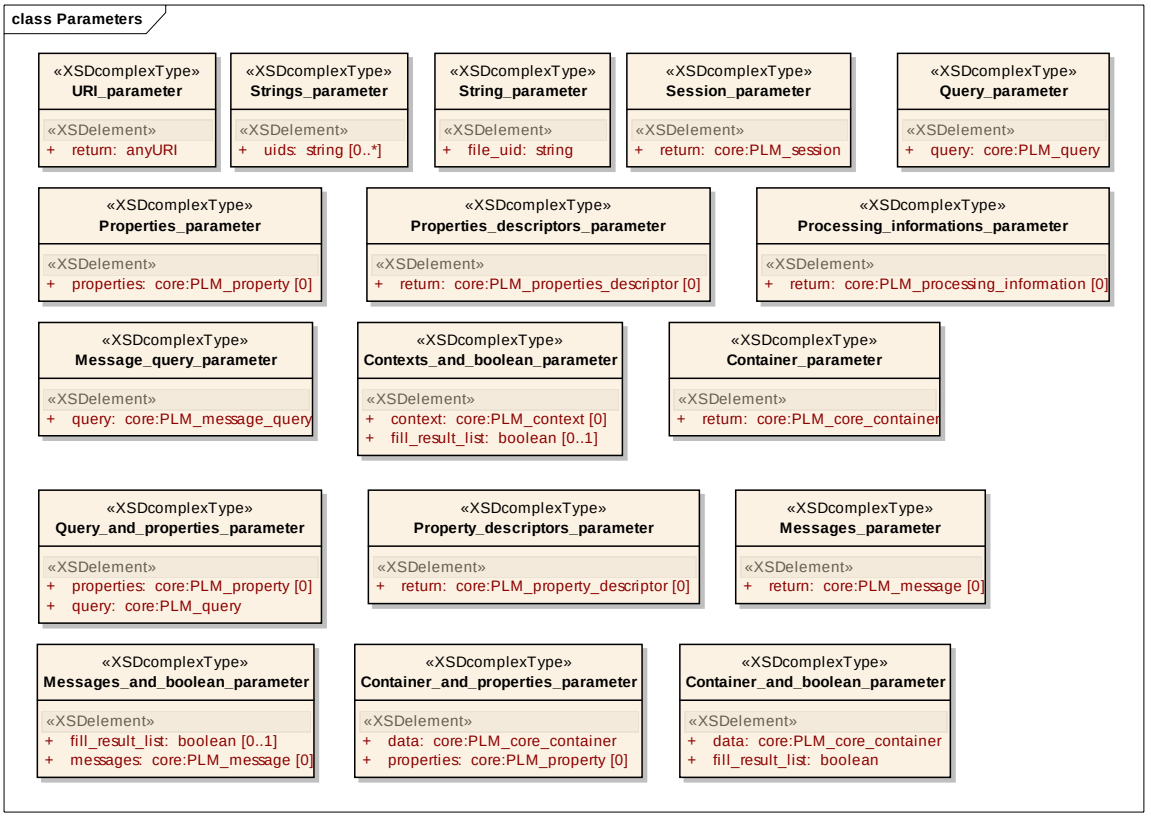


**Figure 10.6 - Parameters**

### 10.4.6 PDTnet Queries

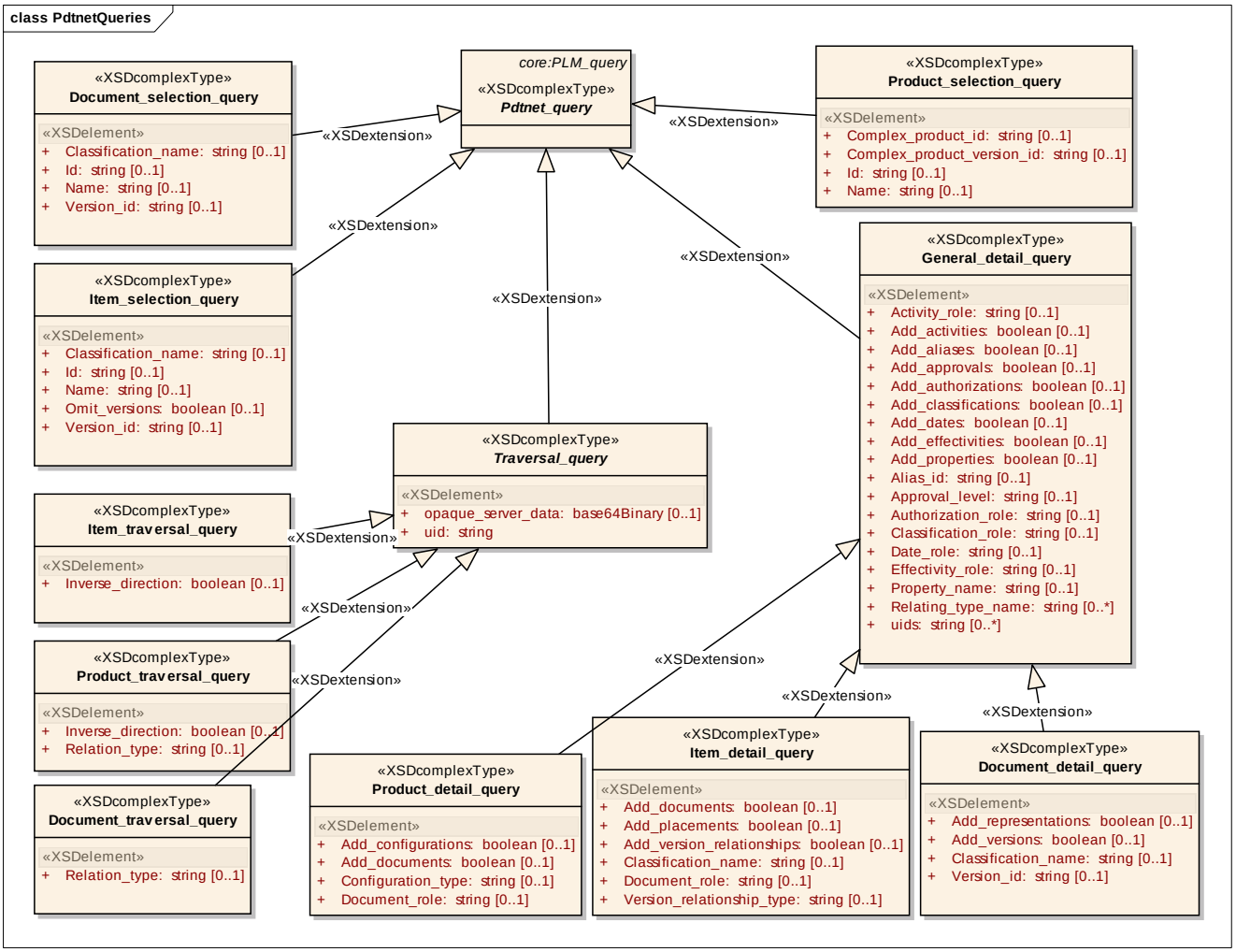


**Figure 10.7 - PDTnet Queries**

### 10.4.7 Proxy Queries

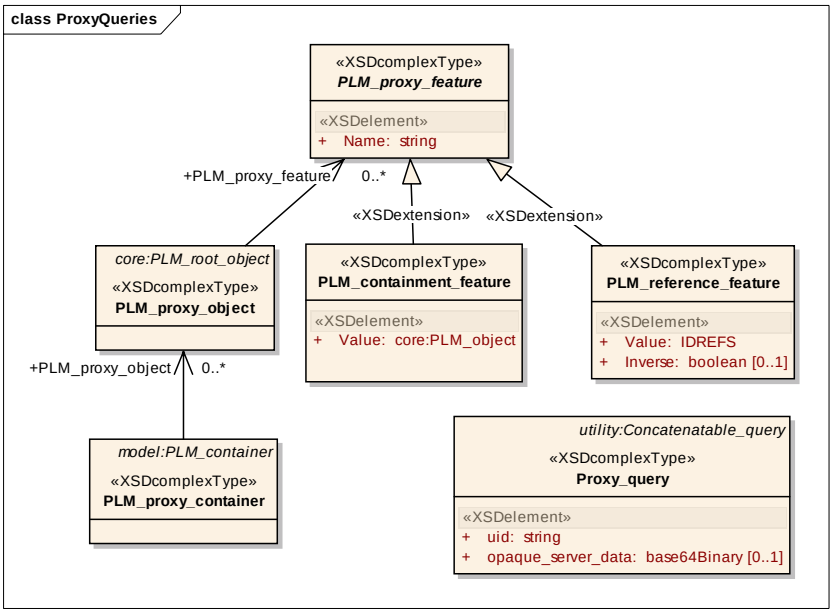


**Figure 10.8 - Proxy Queries**

### 10.4.8 Schema Infos

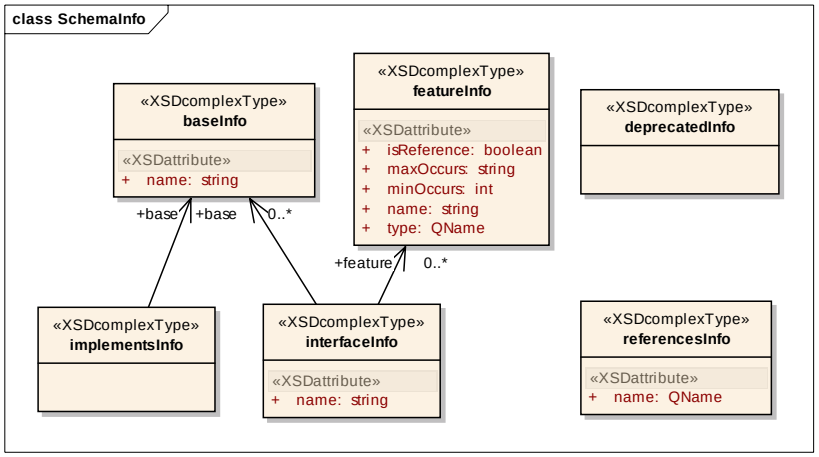


**Figure 10.9 - Schema Info**

### 10.4.9 Specific Queries

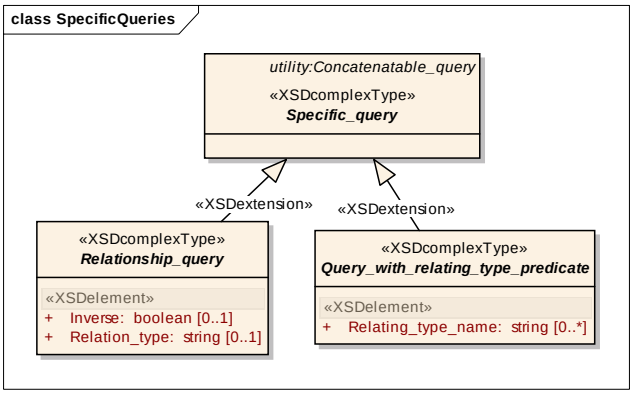


**Figure 10.10 - Specific Queries**

### 10.4.10 Utility Queries

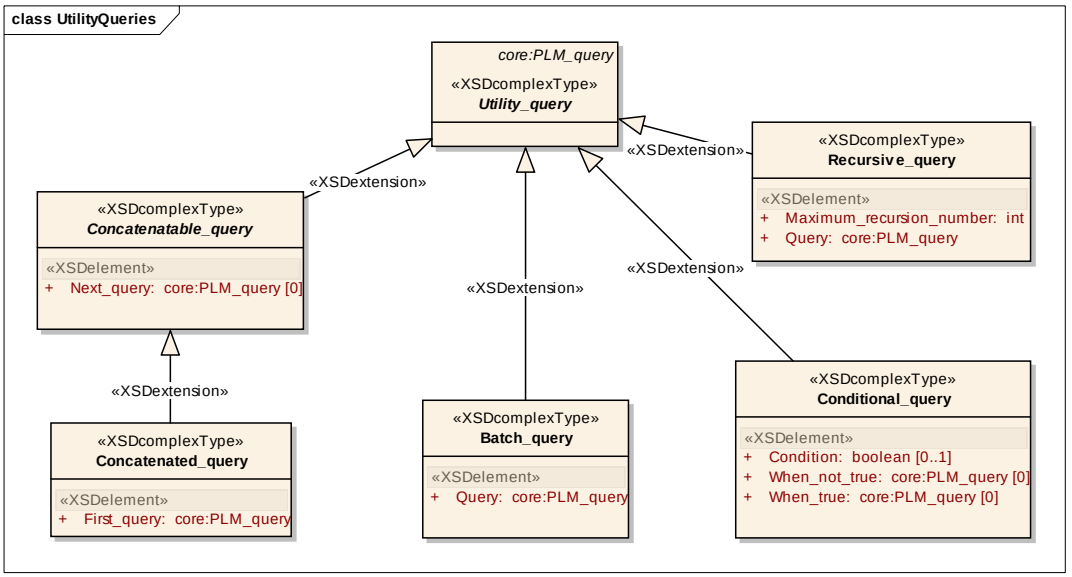


**Figure 10.11 - Utility Queries**

### 10.4.11 XPath Queries

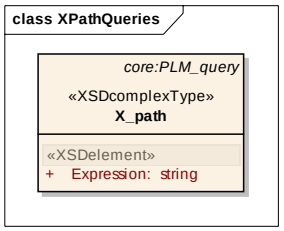


**Figure 10.12 - XPath Queries**

## 10.5 PLM Services Web services (normative)

The PLM Services Computational Viewpoint with applied stereotypes is given in a separate OMG document in XMI which is normative. Additional informative description is given below.

The Computational Viewpoint of the Web service PSM is defined in the Web Services Description Language (WSDL) 1.1. The WSDL imports the XML Schema defined by the Informational Viewpoint. The Web service PSM contains definitions of two ports: PLM_connection_factory and PLM_connection. Due to the fact that Web services can not transfer object references as parameters or results of operations, the syntax and semantic of the operation get_connection() has changed in comparison with the PIM. In the Web service PSM get_connection returns a PLM_session instance which contains a Session_context and a Location element. The Session_context identifies a session and has to be added as a soap header element to each operation request to a PLM_connection port for this session. The optional Location element overrides the address element of the PLM_connection port in the WSDL. The PIM object types PLM_resource_adapter and PLM_object_factory have no counterpart in the Web service PSM.

### 10.5.1 Connection Factory

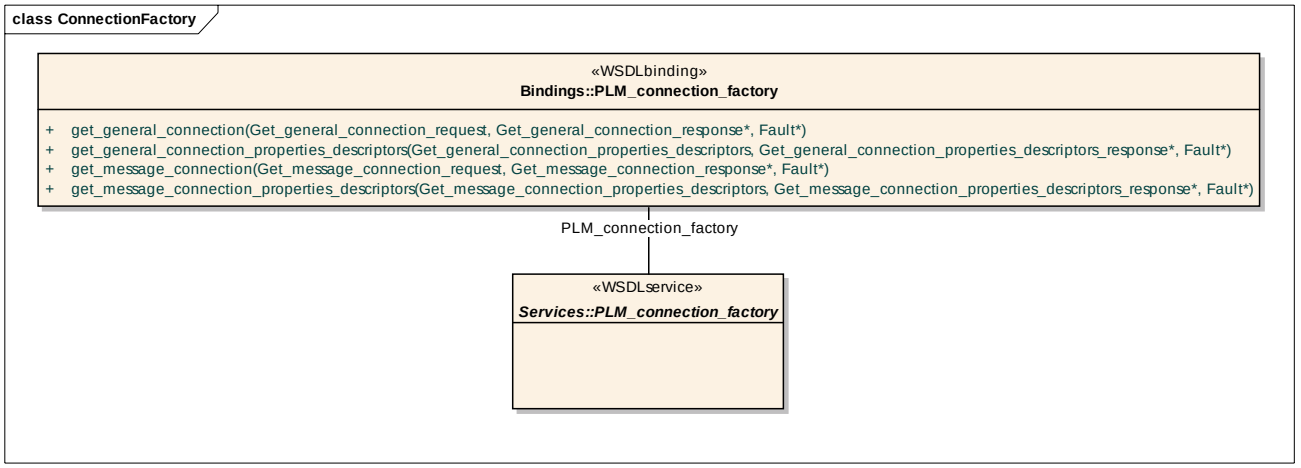


**Figure 10.13 - Connection Factory**

### 10.5.2 General Connection

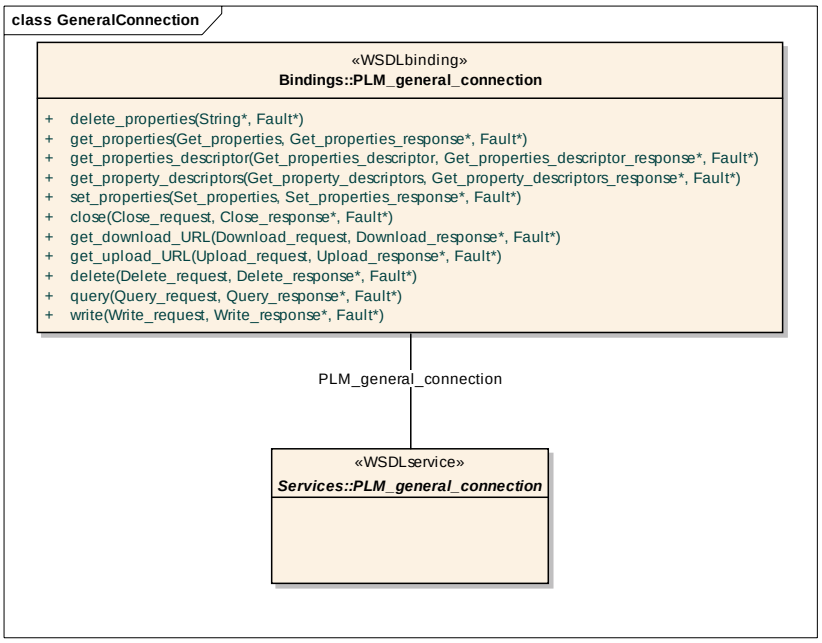


**Figure 10.14 - General Connection WSDL Binding**

### 10.5.3 Message Connection

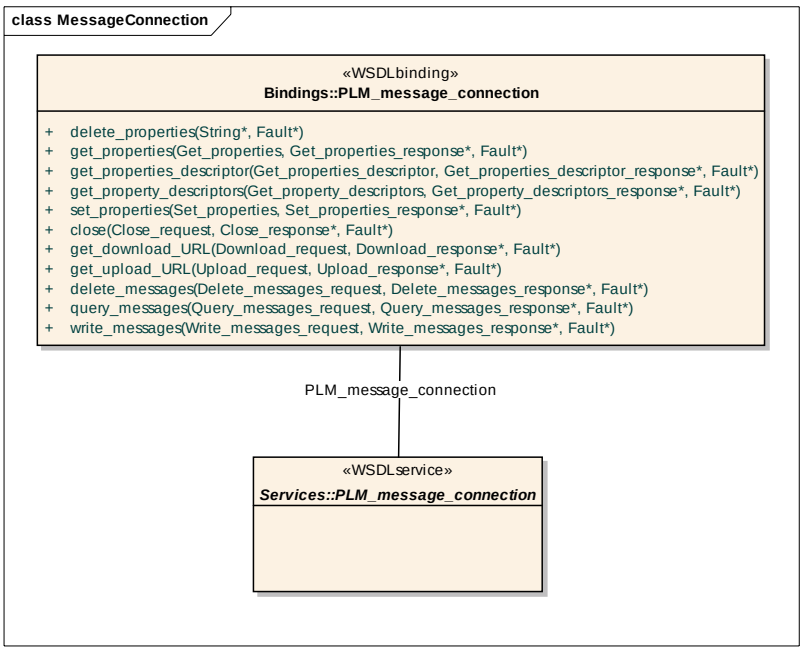


**Figure 10.15 - Message Connection WSDL Binding**

## 10.6 Query Examples (informative)

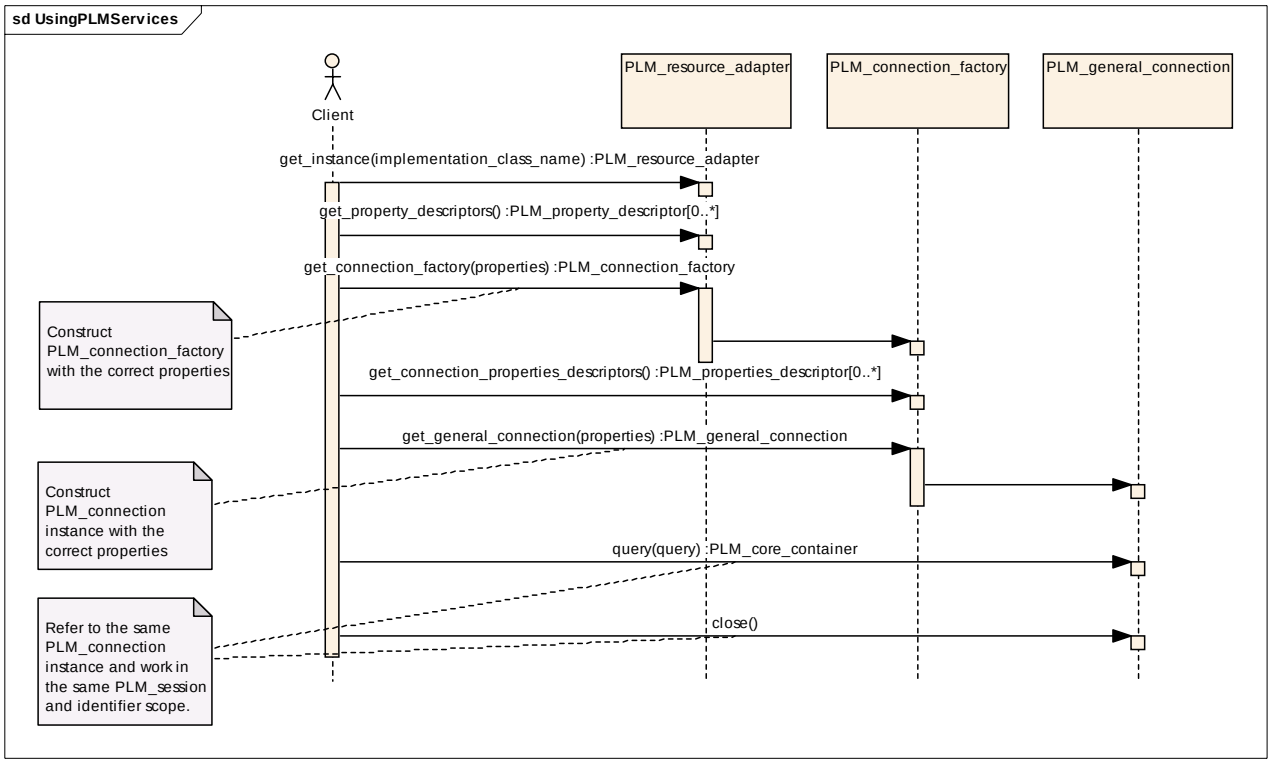


**Figure 10.16 - Sequence diagram of a PLM session**

### 10.6.1 Generic Queries Conformance Point Example

Query for all Item_version objects with id='bar' of Item objects with Name='foo'.

```
<Query xsi:type="Location_path">
    <First_step>
        <Role_name>item</Role_name>
        <Predicate xsi:type="Attribute_predicate">
            <Attribute_name>name</Attribute_name>
            <Attribute_value>foo</Attribute_value>
        </Predicate>
        <Next_step>
            <Role_name>item_version</Role_name>
            <Predicate xsi:type="Attribute_predicate">
                <Attribute_name>id</Attribute_name>
                <Attribute_value>bar</Attribute_value>
            </Predicate>
        </Next_step>
    </First_step>
</Query>
```

### 10.6.2 XPath Queries Conformance Point Example

Query for all Item_version objects of Item objects with id='foo'.

```
<Query xsi:type="X_path">
    <Expression>//Item[Id='foo']/Item_version</Expression>
</Query>
```

### 10.6.3 PDTnet Queries Conformance Point Examples

Assembly_structure_query for all Design_discipline_item_definition objects of Item_version objects with id='4711' of Item objects with name='bar' and name language='en-US'

```
<Query xsi:type="Item_query">
    <Name>bar</Name>
    <Name_language>en-US</Name_language>
    <Next_query xsi:type="Item_version_query">
        <Id>4711</Id>
        <Next_query xsi_type="Design_discipline_item_definition">
            <Next_query xsi:type="Assembly_structure_query"/>
        </Next_query>
    </Next_query>
</Query>
```

Assembly_structure_query for Design_discipline_item_definition with an initial_context with application_domain='mechanical design' and life_cycle_stage='design' of all Item_version objects of Item objects with id='foo'. The result is extended by associated Date_time, Organization and Property_value objects.

```
<Query xsi:type="Item_query">
    <Id>foo</Id>
    <Next_query xsi:type="Item_version_query">
        <Next_query xsi_type="Design_discipline_item_definition">
            <Application_domain>mechanical design</Application_domain>
            <Life_cycle_stage>mechanical design</Life_cycle_stage>
            <Next_query xsi:type="Assembly_structure_query">
                <Next_query xsi:type="Associated_date_time_query"/>
                <Next_query xsi:type="Associated_organization_query"/>
                <Next_query xsi:type="Associated_property_query"/>
            </Next_query>
        </Next_query>
    </Next_query>
</Query>
```

Assembly_structure_query for the PLM_object with uid='assembly123' (which should be an Assembly_definition).

```
<Query xsi:type="Object_by_uid_query">
    <uid>assembly123</uid>
    <Next_query xsi:type="Assembly_structure_query"/>
</Query>
```

## 10.7 Security Features (informative)

### 10.7.1 Introduction

The intent is to foster the use of existing standards for security within SOAP communication and not to define a new security standard to enable secure communication between a sender and recipient.

### 10.7.2 Goals and Requirements

The security features defined here are based on [WS-SECURITY].

The WS-Security standard provides a flexible set of mechanisms to enable SOAP message security together with a high level of interoperability. Therefore this standard offers the possibility to integrate further WS-Security standards and other security standards like SAML etc., which are based on the WS-Security Header. The WS-Security standard also describes mechanisms to use XML-Encryption and XML-Signature, which are W3C (World Wide Web Consortium) standards to encrypt and sign XML-Documents.

### 10.7.3 Non-Goals

The following topics are outside the scope of this document:

Key derivation or key management

Security policies

### 10.7.4 Authentication and Authorization Security

This describes the secure exchange of authentication and authorization information (rather than the mechanisms for authentication or authorization itself) through the use of security tokens described in the WS-Security specification. These security tokens in combination with WS-Security enable a standardized way to place security related data in the SOAP header. This data can be a username and password directly placed in the SOAP header or a reference to security related data in the SOAP body.

The security related data are placed within the <wsse:Security> element in the SOAP-Header. Doing so, all receivers can identify and handle this security information. For more details, refer to [TOKEN].

#### 10.7.4.1 Usage of Security Tokens (UsernameToken)

The UsernameToken transports standard data (like username and password) within the WS-Security header for user authentication. It shall be mandatory when the consumer of the “PLM_connection_factory” calls the method “get_connection”. Doing so, the “PLM_connection_factory” can identify the requestor by username and password before returning a connection.

***Associated Changes in the PLM Services Specification***

*The use of the UsernameToken changes the OMG PLM Services 1.0 specification in chapters 8.3 “PLM_property_descriptor and PLM_properties_descriptor” and 8.3.1 “Sample ‘login’ PLM_properties_descriptor”. It is not allowed anymore to place the username and password in the “PLM_properties_descriptor” (i.e. for authentication). The “PLM_properties_descriptor” shall be used only for additional information. Additional information means information required for authorization or other needs.*

*The usage of the method “get_properties_descriptors” in the “PLM_connection_factory” must also be extended with the UsernameToken(username and password). Without the implementation of the UsernameToken it is not possible to pass an XML firewall to call the “PLM_connection_factory”.*

The following examples or descriptions are partially from [TOKEN]

**XML-Schema**

```
<xs:element name="UsernameToken">
  <xs:complexType>
    <xs:sequence>
```

```
        <xs:element ref="Username"/>
        <xs:element ref="Password" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:ID"/>
      <xs:anyAttribute namespace="##other"/>
   </xs:complexType>
</xs:element>
```

The password contains an attribute „type“, which specifies the type of password being provided. This can be plain text or digest.

**Textformat**

```
<wsse:UsernameToken>
   <wsse:Username>Sven</wsse:Username>
   <wsse:Password Type="wsse:PasswordText">Kennwort</wsse:Password>
</wsse:UsernameToken>
```

It is also possible to encrypt the password or the whole UsernameToken by using the xml encryption mechanisms provided by WS-Security.

The “nonce” and “created” can be used to define the “PasswordDigest”. Proper instructions how to use the “nonce” and “created” in combination with the SHA1(Secure Hash Algorithm 1) to form the “PasswordDigest” can be found in [TOKEN].

**Digestformat**

```
<wsse:UsernameToken>
   <wsse:Username>Sven</wsse:Username>
   <wsse:Password Type="wsse:PasswordDigest">
      KE6QugOpkPyT3Eo0SEgT30W4Keg=</wsse:Password>
   <wsse:Nonce>5uW4ABku/m6/S5rnE+L7vg==</wsse:Nonce>
   <wsu:Created xmlns:wsu=
      "http://schemas.xmlsoap.org/ws/2002/07/utility">
         2002-08-19T00:44:02Z
   </wsu:Created>
</wsse:UsernameToken>
```

Extract from [TOKEN]:

/wsse:UsernameToken/wsse:Password/@{any}

This is an extensibility mechanism to allow additional attributes, based on schemas, to be added to the element.

/wsse:UsernameToken/wsse:Nonce

This optional element specifies a cryptographically random nonce. Each message including a <wsse:Nonce> element MUST use a new nonce value in order for web service producers to detect replay attacks.

/wsse:UsernameToken/wsse:Nonce/@EncodingType

This optional attribute URI specifies the encoding type of the nonce (see the definition of <wsse:BinarySecurityToken> for valid values). If this attribute isn't specified then the default of Base64 encoding is used.

/wsse:UsernameToken/wsu:Created

The optional <wsu:Created> element specifies a timestamp used to indicate the creation time. It is defined as part of the <wsu:Timestamp> definition.

### 10.7.4.2 Examples

The following example shows a full SOAP-Envelope with a <wsse:Security> element containing a UsernameToken. This security header shows the minimum required security header for PLM Services.

**UsernameToken without password encryption**

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Header>
         <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="1">
            <wsse:UsernameToken>
               <wsse:Username>test</wsse:Username>
               <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        username-token-profile-1.0#PasswordText">security</wsse:Password>
            </wsse:UsernameToken>
               …
         </wsse:Security>
      </soapenv:Header>
      <soapenv:Body>
        …
      </soapenv:Body>
```

```
</soapenv:Envelope>
```

The next example shows the same security header with an encrypted password within the UsernameToken.

**UsernameToken with password encryption**

```
<?xml version="1.0" encoding="UTF-8"?>

   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"

        xmlns:xsd="http://www.w3.org/2001/XMLSchema"

        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

      <soapenv:Header>

        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-

        wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="1">

            <xenc:EncryptedKey>

               <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-

        1_5"></xenc:EncryptionMethod>

               <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

                  <wsse:SecurityTokenReference>

                     <wsse:KeyIdentifier EncodingType="http://docs.oasis-

open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://
docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3">MIIBYjCCAQygAwIBAgIQIWFO9wjTxZJOxcgGtBqGVTANBgkqhkiG9w0BAQQFADAPMQ0wCwYDVQQDEwRkaW1zMB4XDTAzM
DUxMjE2NDExN1oXDTM5MTIzMTIzNTk1OVowDzENMAsGA1UEAxMEZGltczBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQDrmZ7T2MFQNwloGu
ghSRoapkmvbtPAwBXt+21bFzqfXJ1SpliN6CCRczIflSQCCCyBZ2j0dA51n/
ZDWDizdNenAgMBAAGjRDBCMEAGA1UdAQQ5MDeAEBsIiVESxf6DrjkLYXayxmKhETAPMQ0wCwYDVQQDEwRkaW1zghAhYU73CNPFkk7FyAa
0GoZVMA0GCSqGSIb3DQEBBAUAA0EAxSGwjZ/FOScVLlVTxic1FKmPd8WTg1DrJFDWuxMTx6n0Zxn4N8ZxkAl7TNx/
JcIlG+dlnyWZ0in3dOEtF0g5mA==

                     </wsse:KeyIdentifier>

                  </wsse:SecurityTokenReference>

               </ds:KeyInfo>

               <xenc:CipherData>
```

```
        <xenc:CipherValue>4eJCKfqXwrkYddfHcO0j/ZQ8q3pMbTk+Wo3AQSwNboDp7/ej8kUSR6F4eM7Ld
        14lhh1gNR7JrDKwE4fq2RPaNg==</xenc:CipherValue>
                </xenc:CipherData>
                <xenc:ReferenceList>
                   <xenc:DataReference URI="#EncDataId-21721154"></xenc:DataReference>
                </xenc:ReferenceList>
             </xenc:EncryptedKey>
             <wsse:UsernameToken>
                <wsse:Username>test</wsse:Username>
                <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        username-token-profile-1.0#PasswordText">
                   <xenc:EncryptedData Id="EncDataId-21721154"
        Type="http://www.w3.org/2001/04/xmlenc#Content">
                      <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"></xenc:EncryptionMethod>
                      <xenc:CipherData>
                         <xenc:CipherValue>YwFBdy5F2xFj5Y8wOYjmAB7vW8obeSmI</xenc:CipherValue>
                      </xenc:CipherData>
                   </xenc:EncryptedData>
                </wsse:Password>
             </wsse:UsernameToken>
          </wsse:Security>
       </soapenv:Header>
       <soapenv:Body>
                …
       </soapenv:Body>
```

```
</soapenv:Envelope>
```

### 10.7.5 Data-Exchange Security

The WS-Security standard enables the sender to encrypt or sign the SOAP message, independently if the information which should be signed or encrypted is in the header or the body of the SOAP message.

The encryption or signing of data sent by the client or provider is optional. All information need by a receiver of an encrypted or signed message is contained in the security header. Only the key that was used for the encryption is not placed in the security header. The key management between interoperating parties is not defined within this specification.

Any data within the SOAP body may be encrypted or signed. Which data shall be encrypted or signed depends on the concrete application scenario.

For more details, refer to [SEC].

The encryption of data secures the data over several intermediaries. This means the data sent from one Web Service to another shall not be readable for intermediate servers, but only for the receiver specified with the attribute ‘role’/’actor’ (depending on the used SOAP version). For more information about the XML-Encryption standard refer to the XML-Encryption specification at the W3C organisation on [ENC].

**Example for actor:**

```
<?xml version="1.0" encoding="UTF-8"?>

   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"

        xmlns:xsd="http://www.w3.org/2001/XMLSchema"

        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

      <soapenv:Header>

         <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-

               wssecurity-secext-1.0.xsd" soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
        soapenv:mustUnderstand="1">

               …

         </wsse:Security>

      </soapenv:Header>

      <soapenv:Body>

               …

      </soapenv:Body>
```

```
</soapenv:Envelope>
```

XML-Signature specification is also a W3C standard and defines a XML writing for digital signatures. It enables the sender to sign any data within a XML document. If embedded in the SOAP document, the XML Signature is called an enveloped signature. The hash code of the signature can be used to prove that the message hasn't been manipulated on his way from the sender to the receiver. So the receiver can check the authenticity of the sender and of the message. For more details about the XML-Signature standard refer to the XML- Signature specification at the W3C organisation on [SIGNATURE].

#### 10.7.5.1 Attachements Exchange Security

The WS-Security standard enables the sender to encrypt or sign the SOAP message. The encryption allows encryption on any part of the SOAP message, the header, the body an although the possibly added attachments. PLM Services 2.0 does not describe how to secure a attachment transfer by a separate specified URL but there are some recommendations about this.

- use SSL,
- use basic authentication, or
- check transaction context to be sure requestor works within already secured PLM session.

#### 10.7.5.2 SAML

SAML (Secure Assertion Markup Language) is also an OASIS standard. It enables the transport of authentication and authorization information in a SAML assertion. This information as well as the following is only a preview for the further security invention in PLM-Services. The usage of SAML is optional.

WS-Security outlines the basics for using further security standard in a SOAP message. The SAML assertions are also placed in the <wsse:Security> element. This assertion will be generated by a SAML authority which can be implemented by every company.

The process starts with a SAML request that asks for authentication or authorization. The SAML authority generates a SAML assertion with the given information. This requires that the SAML authority uses an existing system to do the authentication or authorization. As a result, the SAML authority generates the mentioned assertion which can contain information about a Subject (authentication, attributes and authorization). This assertion can be send to the client, within the security header of the SOAP message. The client can now use the SAML assertion in every request to assert his authentication or authorization over cross-company borders.

SAML offers therefore a possibility for single sign on and a standardized envelope for security related data.

For further details about SAML refer to the specification and documents at OSAIS on http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security [OASIS].

# A PIM and PSM for Product Lifecycle Management Services support information

## A.1 Mapping Specification (informative)

The mapping of the AIM representation of ISO 10303 AP214 CC21 to the PLM Equivalence Model defined in Section A.2 is given for informative purposes. It is defined in EXPRESS-X. It is informative and not included in this document. The machine readable file is to be found as indicated in Table 1.

## A.2 PIM Equivalence Model (informative)

The platform independent equivalence model is defined in EXPRESS. This model is produced by the EXPRESS-X mapping specification as defined in Section A.1and is equivalent to the CC21 ARM model of ISO10303 AP214. For a documentation see the explanation to the corresponding UML elements in Section 8. The model file is informative and not included in this document. The machine readable file is to be found as indicated in Table 1.

## A.3 PIM models (normative)

The PIM model for Product Lifecycle Management Services is defined in XMI and provided in extra OMG documents. The Informational Viewpoint and the Computational Viewpoint are provided in two separate XML documents. Both are normative and not included in this document. The machine readable files are to be found as indicated in Table 1.

## A.4 PSM models (normative)

The PSM models for Product Lifecycle Management Services are defined in XMI and provided in extra OMG documents. The Informational Viewpoint and the Computational Viewpoint are provided in two separate XML documents. Both are normative and not included in this document. The machine readable files are to be found as indicated in Table 1.

## A.5 PSM (normative)

The PSM for Product Lifecycle Management Services is defined in XML Schema and in WSDL and provided in extra OMG documents. The Informational Viewpoint (XML Schema) and the Computational Viewpoint (WSDL) are provided in two separate file systems. Both are normative and not included in this document. The machine readable files are to be found as indicated in Table 1.

# Index