UML model reader's guide

SX004G-B6865-0X000-00

Issue No. 2.0

S-Series IPS specifications
Block release 2021

Publishers:

AeroSpace and Defence Industries Association of Europe

Aerospace Industries Association of America (AIA)

Copyright and user agreement

1 Copyright

All correspondence and queries should be directed to:

ASD
Rue du Trône
1050 Brussels
Belgium

2 Agreement for use of the specification SX004G™ suite of information

2.1 Definitions

SX004G™ suite of information means, but is not limited to:

- the UML reader's guidance SX004G
- any other software or information on the page titled " SX004G Downloads" on the website www.sx000i.org

Copyright holder means AeroSpace and Defense Industries Association of Europe (ASD).

2.2 Notice to user

By using all or any portion of **SX004G™ suite of information** you accept the terms and conditions of this user agreement.

This user agreement is enforceable against you and any legal entity that has obtained **SX0004G™ suite of information** or any portion thereof and on whose behalf, it is used.

2.3 License to use

As long as you comply with the terms of this user agreement, the copyright holders grant to you a non-exclusive license to use **SX004G™ suite of information**.

2.4 Intellectual property rights

SX004G™ suite of information is the intellectual property of and is owned by the copyright holder. Except as expressly stated herein, this user agreement does not grant you any intellectual property right in the **SX004G™ suite of information** and all rights not expressly granted are reserved by the copyright holder.

2.5 No modifications

You must not modify, adapt or translate, in whole or in part, **SX004G™ suite of information**.

2.6 No warranty

SX004G™ suite of information is being delivered to you "as is". The copyright holder does not warrant the performance or result you may obtain by using **SX004G™ suite of information**. The copyright holder makes no warranties, representations or indemnities, express or implied,

whether by statute, common law, custom, usage or otherwise as to any matter including without limitation merchantability, integration, satisfactory quality, fitness for any particular purpose, or non-infringement of third parties rights.

2.7 Limitation of liability

In no event will the copyright holder be liable to you for any damages, claims or costs whatsoever or any consequential, indirect or incidental damages, or any lost profits or lost savings or for any claim by a third party, even if the copyright holder has been advised of the possibility of such damages, claims, costs, lost profits or lost savings.

2.8 Indemnity

You agree to defend, indemnify, and hold harmless the copyright holder and its parents and affiliates and all of their employees, agents, directors, officers, proprietors, partners, representatives, shareholders, servants, attorneys, predecessors, successors, assigns, and those who have worked on the preparation, publication or distribution of the **SX004G™ suite of information** from and against any and all claims, proceedings, damages, injuries, liabilities, losses, costs, and expenses (including reasonable attorneys' fees and litigation expenses), relating to or arising from your use of the **SX004G™ suite of information** or any breach by you of this user agreement.

2.9 Governing law and arbitration

This user agreement will be governed by and construed in accordance with the laws of the Kingdom of Belgium.

In the event of any dispute, controversy or claim arising out of or in connection with this user agreement, or the breach, termination or invalidity thereof, the parties agree to submit the matter to settlement proceedings under the ICC (International Chamber of Commerce) ADR (Alternative Dispute Resolution) rules. If the dispute has not been settled pursuant to the said rules within 45 days following the filing of a request for ADR or within such other period as the parties may agree in writing, such dispute shall be finally settled under the rules of arbitration of the International Chamber of Commerce by three arbitrators appointed in accordance with the said rules of arbitration. All related proceedings should be at the place of the ICC in Paris, France.

The language to be used in the arbitral proceedings shall be English.

Note

The following letter from the Secretary General of ASD extends the special usage rights of this specification and has been included for information, without affecting the technical contents. The content of this letter will be included in the copyright information as part of the S-Series 2024 block release.

ASD AeroSpace and Defence Industries Association of Europe

Special Usage Rights for the S-Series Specifications

To whom it may concern,

This letter seeks to clarify the special usage rights for the suite of documents known as the S-Series Specifications, for which the Aerospace and Defence Industries Association of Europe (ASD) holds the copyright and trademark.

The special usage rights are described as follows:

Permission to use or deliver training from the information contained in the S-Series Specifications, and the right to reproduce or publish the S-Series Specifications, in whole or in part, is hereby given to the following:

1. National Associations who are members of ASD and all their member companies;
2. Members of Aerospace Industries Association of America;
3. Members of the ATA e-Business Program;
4. Members of International Coordinating Council of Aerospace Industries Associations (ICCAIA) not included in categories 1 through 2 inclusively;
5. Airlines and Armed Forces that are customers of Companies included in Categories 1 through 3 inclusively;
6. Ministries of Defence of the member countries of ASD, of NATO and NATO Partners[1];
7. The Department of Defense of the USA;
8. NATO bodies, organizations & agencies;
9. Universities;
10. Technologies and Research Institutes.

Any further requirement for clarification, should in the first instance be directed to the Service Commission of the ASD.

Yours Sincerely,

Jan Pie

Secretary General of ASD

[1] as per: https://www.nato.int/cps/en/natohq/51288.htm);

Highlights

Table of contents

Page

List of tables

References

Table 1 References

Chap No./Document No.	Title
SX002D	Common data model for the S-Series IPS specifications

1 General

Changes that are included in this Issue 2.0 are brought about by changes to the UML Modeling Style used for all S-Series Integrated Product Support (IPS) specifications where its data models are based on SX002D Common Data Model Issue 2.0. The combined result is that most illustrations are updated. There are also some additional concepts and constructs that have been added to Issue 2.0, that did not exist in the UML Modeling Style used for S-Series IPS specifications where its data models were based on SX002D Common Data Model Issue 1.0 and Issue 1.1.

1.1 Changes

New chapter structure for Chap 2 where:

- Para 1 now only contains General information
- Para 2 describes how to read UML Class models using the S-Series IPS specifications UML profile
- Para 3 highlights some very specific S-Series IPS specifications Class model style decisions
- Para 4 describes the respective data type used in the UML Class models

UML Class attributes are described in a separate Para (refer to Chap 2 Para 2.3).

Former Para on association relationships now also explains the use of `<<relationship>>` Classes in the context of association relationships (refer to Chap 2 Para 2.5).

Former Para on directed associations is deleted.

Description of `local` associations is added (refer to Chap 2 Para 3.2).

Added descriptions on new S-Series IPS specifications `<<primitive>>` data types `NameType` and `StateType` (refer to Chap 2 Para 4.2.6 and Para 4.2.8).

Added graphical representation of S-Series IPS specifications `<<primitives>>` data types (refer to Chap 2 Para 4.2.9).

Added descriptions on new S-Series IPS specifications <<umlPrimitives>> data types that matches defined UML primitives (refer to Chap 2 Para 4.3).

Added descriptions on new S-Series IPS specifications <<compoundAttribute>> data types TimeStampedClassification, DateTimeRange and ThreeDimensional (refer to Chap 2 Para 4.4.3, Para 4.4.5 and Para 4.4.7).

Added graphical representation of S-Series IPS specifications <<compoundAttribute>> data types (refer to Chap 2 Para 4.4.8)

Added description on using Organization as the Class attribute data type (refer to Chap 2 Para 4.5).

Added description on validValue as the Class attribute data type (refer to Chap 2 Para 4.6).

Table of contents

The listed documents are included in Issue 2.0 dated 2021-04-30, of this publication.

Chapter 1

Introduction to the specification

Table of contents

Chapter 1.1

Purpose

Table of contents

List of tables

References

Table 1 References

Chap No./Document No.	Title
S1000D	International specification for technical publications using a common source database
SX000i	International specification for Integrated Product Support (IPS)
SX002D	Common data model for the S-Series ILS specifications
S3000L	International procedure specification for Logistic Support Analysis (LSA)
S5000F	International specification for in-service data feedback
S6000T	International specification for training analysis and design
http://www.uml.org	Unified Modelling Language (UML)

1 General

The UML model reader's guide Issue 2.0 for the S-Series Integrated Product Support (IPS) specifications is a document describing how to read and understand the Unified Modelling Language (UML) class models that are created for the S-Series IPS specifications released 2020 and later (eg, S3000L Issue 2.0, S5000F Issue 2.0 and S6000T Issue 1.0), including the common data model. Refer to SX002D Issue 2.0.

2 Purpose

The purpose of the UML model reader's guide is to provide clear instruction on how the S-Series IPS Specification's UML models need to be read by the audience to make sure, all parties have a common understanding.

It assumes the reader has a basic understanding of modelling languages.

The models are described using the UML 2.0™ (Unified Modelling Language) class model diagrams (www.uml.org).

3 Background

The international aerospace and defense community has, over the past 20 years, invested considerable effort developing specifications in the field of IPS. The work is accomplished by integrated working groups composed of industry and customer organizations in a collaborative environment. Customer organizations included representatives from national ministries and departments of defense from Europe and the United States. Aerospace and defense associations provided guidance and supported the work as required. The structure and functional coverage of these specifications was largely determined by North Atlantic Treaty Organization (NATO) requirements specified during an international workshop in Paris in 1993.

Beginning in 2003, the relationships between supporting industry organizations were formalized through a series of Memorandums of Understanding (MOU). Initially AeroSpace and Defense Industries Association of Europe (ASD) and Aerospace Industries Association of America (AIA) signed an MOU to jointly develop and maintain S1000D.

In 2010, ASD and AIA signed an MOU to promote a common, interoperable, international suite of IPS specifications and jointly develop the S-Series IPS specifications. This MOU authorized the formation of the AIA/ASD IPS Specifications Council, whose responsibilities include performing liaison between ASD and AIA, developing and maintaining the S-Series IPS specifications, administering joint meetings and identifying additional areas of harmonization.

The need for a consolidated and harmonized Common Data Model (CDM) was recognized as a fundamental requirement for the complete S-Series IPS specifications. Its creation and maintenance were assigned to the Data Modeling and Exchange Working Group (DMEWG) and is numbered SX002D to align it with SX000i. In order to help the users of S-Series IPS specifications, where its data models are based on the CDM, to understand and read it´s UML constructs, the need for an UML reader's guide was recognized. It is numbered SX004G and created and maintained by the DMEWG as well.

Chapter 1.2

Scope

Table of contents

List of tables

References

Table 1 References

Chap No./Document No.	Title
S1000D	International specification for technical publications using a common source database
S2000M	International specification for material management - Integrated data processing for military equipment
S3000L	International procedure specification for Logistics Support Analysis (LSA)
S4000P	International specification for developing and continuously improving preventive maintenance
S5000F	International specification for in-service data feedback
S6000T	International specification for training analysis and design
SX000i	International specification for Integrated Product Support (IPS)
SX001G	Glossary for the S-Series ILS Specifications
SX002D	Common data model for the S-Series ILS specifications
SX005G	S-Series IPS specifications XML schema implementation guidance
SX006R	S-Series IPS specifications rules definition guide

1 General

SX004G, the UML model reader's guide for the S-Series Integrated Product Support (IPS) specifications is a document describing how to read and understand the UML class models created for the S-Series IPS specifications, including the common data model (SX002D).

2 Scope

This document explains:

- illustrations of UML constructs used within the S-Series IPS specifications
- how to read and interpret the constructs

3 S-Series IPS specifications

The following AeroSpace and Defense Industries Association of Europe (ASD) and Aerospace Industries Association of America (AIA) S-Series IPS specifications are currently available or in the process of development, including:

- S1000D - International specification for technical publications using a common source database
- S2000M - International specification for material management - Integrated data processing for military equipment
- S3000L - International procedure specification for Logistics Support Analysis (LSA)
- S4000P - International specification for developing and continuously improving preventive maintenance
- S5000F - International specification for in-service data feedback
- S6000T - International specification for training analysis and design
- SX000i - International specification for Integrated Product Support (IPS)
- SX001G - Glossary for the S-Series ILS Specifications
- SX002D - Common data model for the S-Series ILS specifications
- SX004G - UML model reader's guide
- SX005G - S-Series IPS specifications XML schema implementation guidance
- SX006R - S-Series IPS specifications rules definition guide

Chapter 1.3

How to use the specification

Table of contents

List of tables

References

Table 1 References

Chap No./Document No.	Title
Chap 1	Introduction to the specification
Chap 2	UML model reader's guide

1 General

This chapter gives an overview of the organization of the specification and the fundamental reading rules.

2 Organization of the specification

2.1 Chapter 1 - Introduction to the specification

Chap 1 provides a summarized view on purpose, background and scope of SX004G.

2.2 Chapter 2 - UML model reader's guide

Chap 2 includes the UML model reader's guidance. The text of Chap 2 includes a Table of Contents with a link to the different modelling constructs used throughout the S-Series UML class models, providing a convenient way to locate a specific term.

Each term entry consists of mandatory and optional components.

3 Harmonization

In 2021, the S-Series Integrated Product Support (IPS) Council decided to harmonize all 14 of the S-Series IPS specifications. To achieve this, an overarching specification has been developed that describes how all the specifications work together, in an integrated manner. The issue date for said specification, (SX000H - Harmonized Suite of the S-Series Integrated Product Support (IPS) Specifications), and all the S-Series IPS specifications was 2021-04-30. Refer to SX000H.

Chapter 1.4

Maintenance of the specification

Table of contents

List of tables

References

Table 1 References

Chap No./Document No.	Title
SX000i	International specification for Integrated Product Support (IPS)
IPS-C-2020-010	Governance of the S-Series Specifications

1 Maintenance of the specification

SX004G is maintained by the Data Modeling and Exchange Working Group (DMEWG) operating under the supervision of the Integrated Product Support (IPS) Specifications Council. Both the DMEWG and the IPS Specifications Council include representatives from AeroSpace and Defense Industries Association of Europe (ASD) and Aerospace Industries Association (AIA) member companies and nations.

Issues related to SX004G can be raised using the change request tool found at www.SX000i.org. Change requests are submitted with the understanding that any revisions to SX004G can affect the other specifications in the S-Series IPS specifications, and that proposed changes are subject to international agreement between ASD and AIA member companies and nations.

Upon receipt of a change request, the DMEWG will follow the change management process described in IPS-C-2020-010, to gain consensus agreement from the participating organizations prior to the publication of changes. The DMEWG considers change proposals and ratifies them for incorporation into SX004G.

Chapter 2

UML model reader's guide

Table of contents

List of tables

List of figures

References

Table 1 References

Chap No./Document No.	Title
S2000M	International specification for material management - Integrated data processing for military equipment
S3000L	International procedure specification for Logistics Support Analysis (LSA)
S4000P	International specification for developing and continuously improving preventive maintenance
S5000F	International specification for in-service data feedback
S6000T	International specification for training analysis and design
SX000i	International specification for Integrated Product Support (IPS)
SX002D	Common data model for the S-Series IPS specifications
DMEWG	Data Model and Exchange Working Group
ISO 10303:239	Product Life Cycle Support (PLCS)
OMG	Object Management Group (www.omg.org)

1 General

1.1 Introduction

This specification is targeted to readers of the S-Series Integrated Product Support (IPS) specifications, which are interested in learning more about how to read and understand the data associated with business processes defined in the respective S-Series IPS specifications but with no previous knowledge about the Unified Modeling Language™ (UML). This specification is an addition to the information provided in chapters titled Data model and Data element list in the respective S-Series IPS specifications.

This specification is therefore a supporting specification to the respective process-oriented S-Series IPS specifications:

- S2000M - International specification for material management - Integrated data processing for military equipment
- S3000L - International procedure specification for Logistics Support Analysis (LSA)
- S4000P - International specification for developing and continuously improving preventive maintenance
- S5000F - International specification for in-service data feedback
- S6000T - International specification for training analysis and design
- SX000i – International specification for Integrated Product Support (IPS)

This issue of SX004G is targeted to those issues of the S-Series IPS specifications where its data models are based on SX002D.

1.2 Objective

The objective for this specification is to give the reader a basic understanding on how to read a UML class model so that the reader then can understand the basics of the data that is defined and described in the respective S-Series IPS specifications.

2 Unified Modeling Language™ (UML)

2.1 Overview

UML is a specification developed by the Object Management Group (OMG) and is a widely used technique to model not only application structure, behavior, and architecture, but also business processes and data structures. It consists of a set of different modelling techniques of which the S-Series IPS specifications use only one, the class model. A class model defines a static view of the information (data) that are needed to support the business processes.

Class models are the most widely used part of UML. They show the business concepts that must be represented as data, and their inter-relationships.

This specification gives an overview of how the UML class model constructs are used in the S-Series IPS specifications data models, based on the UML profile (style) that is defined in the UML Writing Rules and Style Guide published as an internal document by the AeroSpace and Defense Industries Association of Europe (ASD) and Aerospace Industries Association of America (AIA) Data Model and Exchange Working Group (DMEWG).

Note

This chapter does not provide a complete description of UML class modelling techniques, but introduces only those constructs that are relevant to the reader of the S-Series IPS specifications data models.

Some UML class model concepts are also translated into a relational example. These examples are provided for those readers that understand relational databases, but have no previous knowledge of UML and are only intended to give examples of how UML class model concepts can be represented using a relational database.

Note

They must not be interpreted as the solution for an implementation. They are an alternative view to illustrate the concepts.

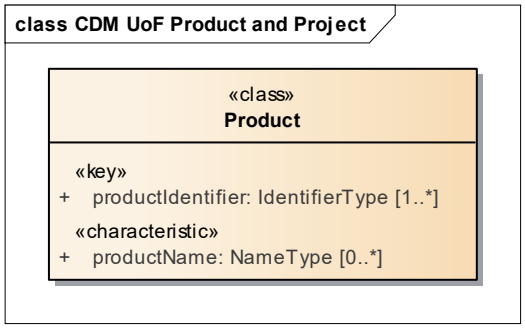
2.2 Class

The basic concept within the S-Series IPS specifications data model diagrams is a box called Class. A Class typically represents a business concept. At the top of the Class box, the name is stated followed by an enumeration of its attributes. Attributes are data elements that identify and describe instances of the Class. An instance of the class is the actual representation (object) of the business concept, while the Class itself can be seen as a template.

Classes which are illustrated with the header <<class>> represent tangible (eg, Product, person, etc) as well as intangible business concepts (eg, environment, learning objective, etc).

This header distinguishes business concept Classes from other usages of the UML Class construct (Refer to <<relationship>> Para 2.3).

A Class can also have relationships to other classes in terms of association, composition, aggregation, generalization/specialization and realization. Refer to Para 2.5 through Para 2.10.



ICN-B6865-SX004G0001-001-01

Fig 1 Example of a <<class>> Class in UML

2.3 Class attributes

A Class can have one or many attributes. Attributes are data elements that identify and describe instances of the Class. Each attribute is presented with its attribute name, data type, and cardinality (multiplicity).

Class attributes are divided into categories (stereotypes in UML). Each category is shown within double angle brackets (<<…>>) above the attribute name(s).

There are five categories of attributes:

- <<key>>
- <<compositeKey>>
- <<relationshipKey>>
- <<characteristic>>
- <<metadata>>

Attributes that are required to uniquely identify a Class instance are classified as either <<key>>, <<compositeKey>> or <<relationshipKey>>. Which type of key attribute that is used is dependent on how the Class is defined in relation to other Classes. Attributes that comprise the key are always defined first in the attribute list for the Class.

The <<key>> category is used for Classes where the only key attributes needed to uniquely identify a Class instance is defined within the Class itself. Refer to Fig 1.

The <<compositeKey>> category is used, when <<key>> attributes from a Class instance also require the <<key>> from a parent composition Class to uniquely identify a Class instance. For example, identification of the Product is not possible with the productVariantIdentifier with a value of "800" or "900". Only when combined with the productIdentifier with a value of "B787" can the Class instance be uniquely identified. Therefore, the productVariantIdentifier of the child composition Class ProductVariant is categorized as <<compositeKey>> to indicate that the productVariantIdentifier must be combined with the productIdentifier <<key>>, defined for the parent composition Class Product, in order to uniquely identify a specific ProductVariant. Refer to Para 2.8.

The <<relationshipKey>> category is used, when there can be more than one instance of the same relationship type between the same relating and related Class instances. Refer to Para 2.5.

The <<characteristic>> category is used for attributes that define the characteristics of a given Class instance (object). Characteristics typically include names, measurable properties, classifications and descriptions.

The <<metadata>> category is used for attributes that define metadata such as the means of creation, author, time and date of creation, etc.

The data type and cardinality for an attribute is shown directly after the attribute name. For allowed data types refer to Para 4. The cardinality (ie, multiplicity) defines how often an attribute can or must be present (refer to Table 2)

Table 2 Cardinalities used in attributes

Cardinality	Explanation
1	One, and only one, instance of the attribute per Class instance (object) must be populated (a mandatory attribute)
0..1	One, and only one, instance of the attribute per Class instance (object) can be populated (an optional attribute)
0..*	Zero, one or many instances of the attribute per Class instance (object) can be populated (an optional attribute, which can have multiple occurrences in the same Class instance)
1..*	At least one instance of the attribute must be populated per Class instance (object) (a mandatory attribute, which can have optional additional occurrences in the same Class instance)

The cardinality for an attribute is defined in the same way as cardinality for an association, which is described in Para 2.5. If there is no explicit cardinality for a given attribute, it means that the attribute must have one value and one value only.

An example of Class attributes is given in Fig 1. It shows the class Product with its two attributes, productIdentifier and productName. Each attribute also shows its group, data type and cardinality.

A Class can have zero, one or many instances and can be seen as the template for its instances. An example on a Product instance is the "New Fighter Aircraft Product". Refer to Fig 2.

A Class in UML can be viewed as a table in a relational database, and an instance of that Class can be seen as a row within that table. The attributes of a Class can be seen as the table columns.

Product	
productIdentifier	**productName**
P-123	New Fighter Aircraft

ICN-B6865-SX004G0002-001-01

Fig 2 Example of possible relational representation of the UML <<class>> Class Product

Note

In the examples, the relational table column that constitutes its primary key is emphasized by underlining the column name(s).

2.4 Class attribute groups

A special case for defining attributes for a Class is the use of attribute groups. Attribute groups are typically used when the list of attributes defined for a Class becomes too complex, and/or when there is some logic that is important to make explicit, (eg, different stakeholders such as "Design" vs "Commercial").

Attribute groups are illustrated using a Class rectangle but with the header `<<attributeGroup>>`. They are always connected to a Class by a composition association (refer to Para 2.8), which is illustrated by a line with a filled diamond connected to the Class for which the attributes are defined.

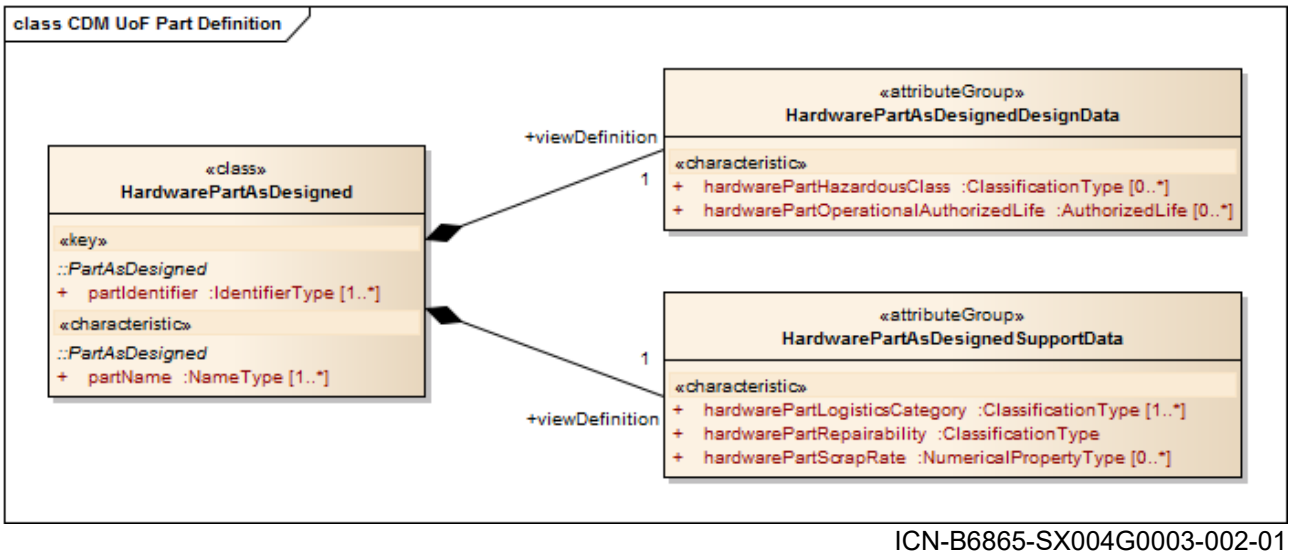
Note

These attributes must be seen as any other attributes defined for the Class it is associated with, and could also have been defined within the parent Class.

Fig 3 shows an example of how the attributes associated with the hardware part (`HardwarePartAsDesigned`) are split in two attribute groups, one relating to the characteristics of a part as relevant for the support (`HardwarePartAsDesignedSupportData`) and the other one as relevant to the design (`HardwarePartAsDesignedDesignData`).

Note

The `HardwarePartAsDesigned` Class in Fig 3 illustrates attributes inherited from its parent class (Refer to Para 2.6) to be consistent with the example in Fig 4.



ICN-B6865-SX004G0003-002-01

Fig 3 Example of an attribute group in UML

An attribute group can be viewed as additional columns within the same table in a relational database, all attached to the same key. Refer to Fig 4.

HardwarePartAsDesigned			
partIdentifier	partName	hardwarePartScrapRate	hardwarePartOperationalAuthorizedLife
128191-219-219	Engine	2 Percent	500 Flight Hours
128491-220-329	Radar	3 Percent	2 Years

ICN-B6865-SX004G0004-001-01

Fig 4 Example of possible relational representation of an attribute group

2.5 Associations and Relationship Class

Associations represent interdependencies between classes. Associations have an open arrowhead at the target end of the association. This arrow indicates the direction for the association (ie, it defines which Class controls the existence of the association). Where both the relating Class and the related Class instances can be part of the defined association more than once then a relationship Class is defined. Such many-to-many associations often require their own characterizations. Relationship classes are modelled as a Class labeled with `<<relationship>>`, with connectors between the two associated classes.

For example, a many-to-many association that is car ownership. A car can be owned by many persons (especially over time) and one person can own more than one car.

An example of an association using a relationship Class is shown in Fig 5. It shows that a hardware part (`HardwarePartAsDesigned`) can be associated with zero, one or many instances of contained substances. At the same time each defined substance (`SubstanceDefinition`) can be contained in zero, one or many hardware parts (`HardwarePartAsDesigned`). The direction for the association is from the part to the substance (ie, it is the part that defines which substances are included, not the opposite). The cardinality (multiplicity) for the association is given at the respective end of the association (Refer to Table 3). The cardinality for an association is defined in the same way as cardinality for an attribute, which is described in Para 2.3.

Table 3 Cardinalities used in associations

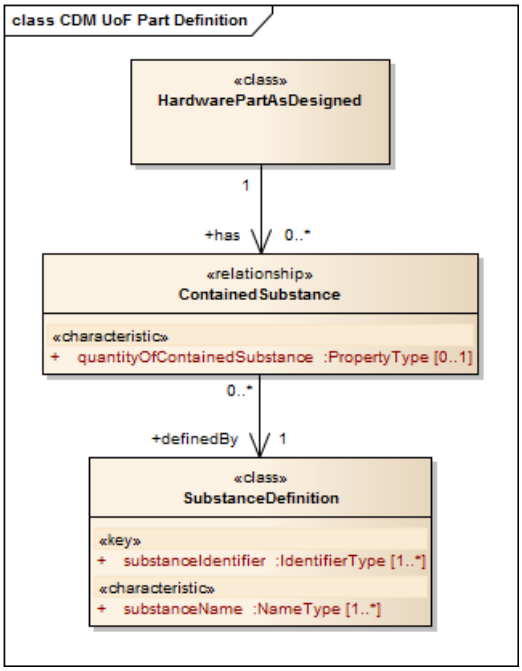
Cardinality	Explanation
1	One (and only one) instance of the associated class must be associated with each instance of the associating class (a mandatory association)
0..1	One (and only one) instance of the associated class can be associated with each instance of the associating class (an optional association)
0..*	Zero, one or many instances of the associated class can be associated with each instance of the associating class (an optional association)
1..*	At least one instance of the associated class must be associated with each instance of the associating class (a mandatory association)

If there is no explicit cardinality for a given association, it means one instance of the associated class must be associated with each instance of the associating class.

Each association between a hardware part (`HardwarePartAsDesigned`) and its contained substance (`SubstanceDefinition`) can have an additional characterization

which defines the quantity of the contained substance. This is enabled by the ContainedSubstance <<relationship>> Class which has the attribute quantityOfContainedSubstance that determines the quantity for a given substance within a given part (eg, 50 gram).

The labels that are attached to the relationship give further indication on the semantics of the relationship and can be read in the direction of the relationship. In the example in Fig 5 one (1) part (HardwarePartAsDesigned) has zero, one or many (0..*) contained substances (ContainedSubstance) with a certain quantity in it and the substance itself is definedBy exactly one (1) substance definition (SubtanceDefinition).



ICN-B6865-SX004G0006-002-01

Fig 5 Example of a <<relationship>> Class in UML

Note

Many-to-many association relationships are defined using explicit <<relationship>> classes, instead of using a simple association directly between the related classes. The rationale for this is that many associations will typically have a set of characteristics that describes them.

An association (including the <<relationship>> Class) in UML can be viewed as a relationship table in a relational database where the columns in the table represent the primary keys from the respective associating and associated Class. Table rows represent references to the associated instances.

Attributes that characterize a <<relationship>> Class can be viewed as additional columns in a relationship table in a relational database (eg, quantityOfContainedSubstance in Fig 6).

The example illustrated in Fig 6 is based on the Class diagram in Fig 5 and shows how two parts (engine and radar) are connected with two substances (quicksilver and gold) that they contain. Furthermore the (optional) information on the quantity of each substance within the part is given in Fig 6.

HardwarePartAsDesigned	
partIdentifier	**partName**
128191-219-219	Engine
128491-220-329	Radar

SubstanceDefinition	
substanceIdentifier	**partName**
XW-PRJAE-9282	Quicksilver
22-PRDAE-9282	Gold

ContainedSubstance		
partIdentifier	**substanceIdentifier**	**quantityOfContainedSubstance**
128191-219-219	XW-PRJAE-9282	2.33 KG
128191-219-219	22-PRDAE-9282	
128491-220-329	XW-PRJAE-9282	0.21 KG

ICN-B6865-SX004G0007-001-01

Fig 6 Example of possible relational representation of an association with additional information

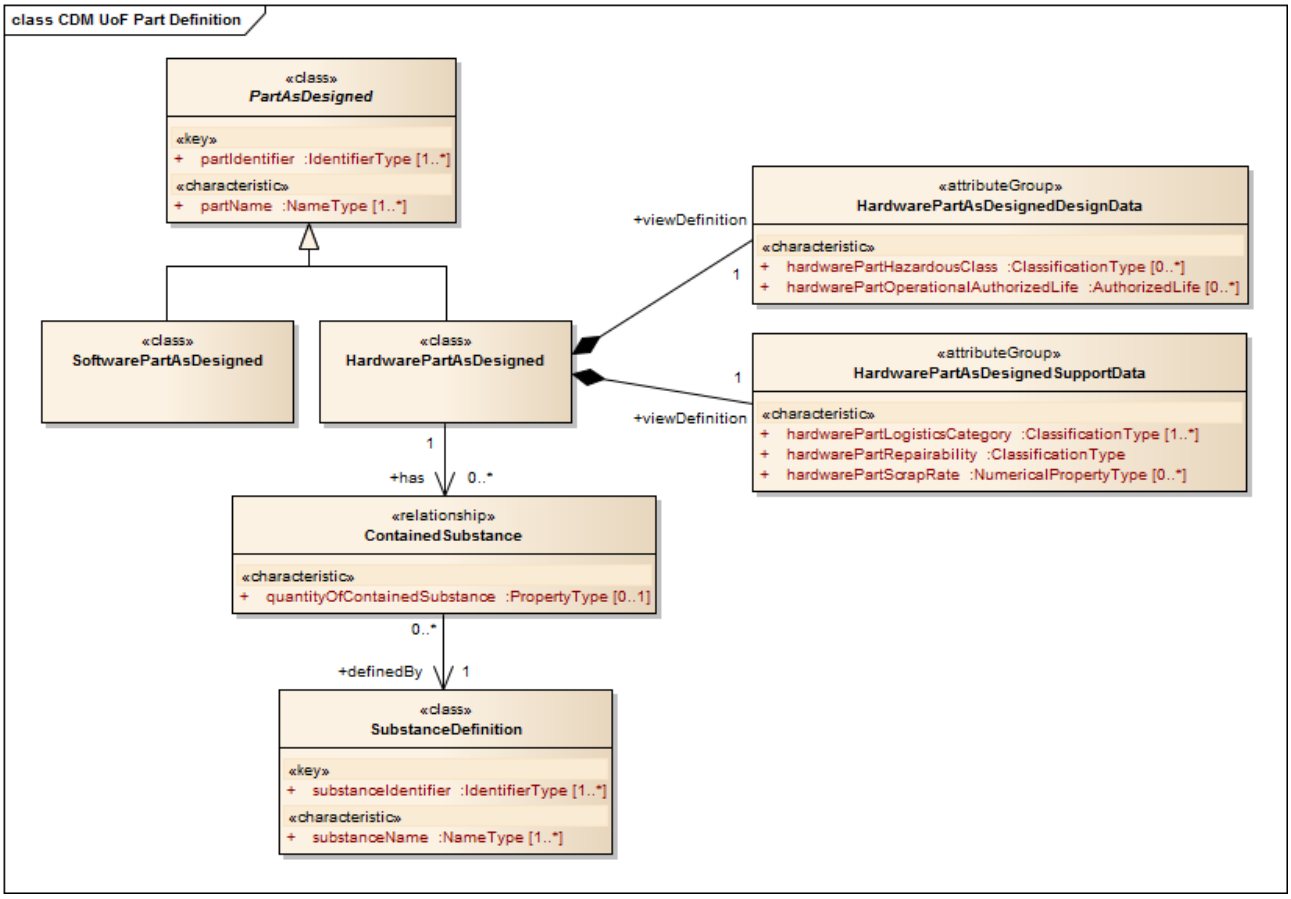
2.6 Generalization and specialization

Specialization refers to a "is-a" relationship and is modelled as a solid line with a hollow triangle at the end that points to the parent Class (ie, the generalized Class). Specialization is often referred to as inheritance.

The most important part of the generalization/specialization relationship is that the child Class gets all the features defined for the parent Class and can then add features of its own, (ie, a child Class inherits all the attributes and relationships defined for its parent Class).

Another important aspect of specialization is substitutability. Substitutability means that wherever a parent Class is being used, a parent can be replaced by any of its children.

An example of the specialization/generalization association is `PartAsDesigned` and its specializations `HardwarePartAsDesigned` and `SoftwarePartAsDesigned`, respectively. In the example shown in Fig 7, the `PartAsDesigned` class has two attributes, `partIdentifier` and `partName`. This means that instances of both `HardwarePartAsDesigned` and `SoftwarePartAsDesigned` will have `partIdentifier` and `partName` as attributes, even though they are not explicitly enumerated in the attribute list for the respective Class. The respective Class then also defines additional attributes which are unique for its respective special characteristics (ie, attributes defined for `HardwarePartAsDesigned` are not applicable to a `SoftwarePartAsDesigned`).



ICN-B6865-SX004G0009-002-01

Fig 7 Example of specialization (inheritance) relationships between classes in UML

Specialization can be viewed as multiple tables with the same initial set of columns (including the <<key>>). Refer to Fig 8.

HadwarePartAsDesigned		
partIdentifier	**partName**	**hardwarePartRepairability**
128191-219-219	Engine	TRUE
128491-220-329	Radar	FALSE

SoftwarePartAsDesigned		
partIdentifier	**partName**	**softwarePartProgrammingLanguage**
128191-219-219	Mission Data System	C++
128491-220-329	Maintenance Data System	C#

ICN-B6865-SX004G0010-001-01

Fig 8 Example of possible relational representation of specializations

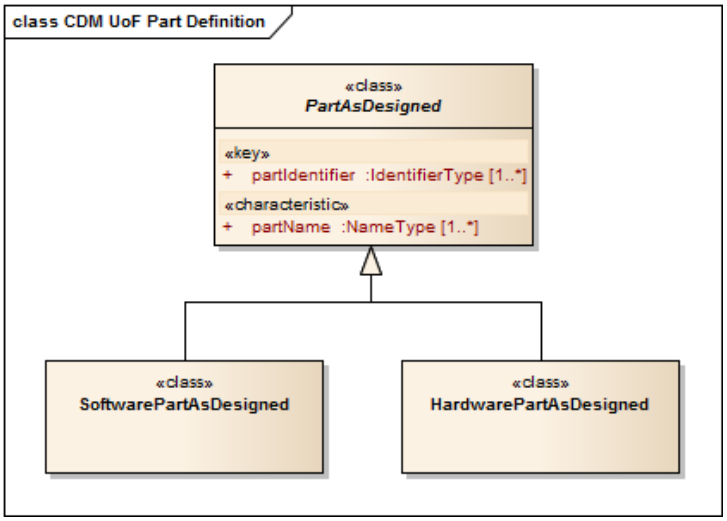
2.6.1 Abstract Class

Classes whose names are written in *italic* are defined as abstract classes in the data model. This means that this Class cannot have any instances (ie, one must always instantiate one of its non-abstract specializations).

In the example shown in Fig 9, `PartAsDesigned` is an abstract Class that cannot be instantiated. One must instead instantiate either of its specializations (ie, instantiate either a `SoftwarePartAsDesigned` or a `HardwarePartAsDesigned`). Both specialization classes inherit the `partIdentifier` and `partName` attributes from their abstract parent `PartAsDesigned`.

Note

A generalized class to be abstract is not required.



ICN-B6865-SX004G0005-002-01

Fig 9 Example of an abstract Class

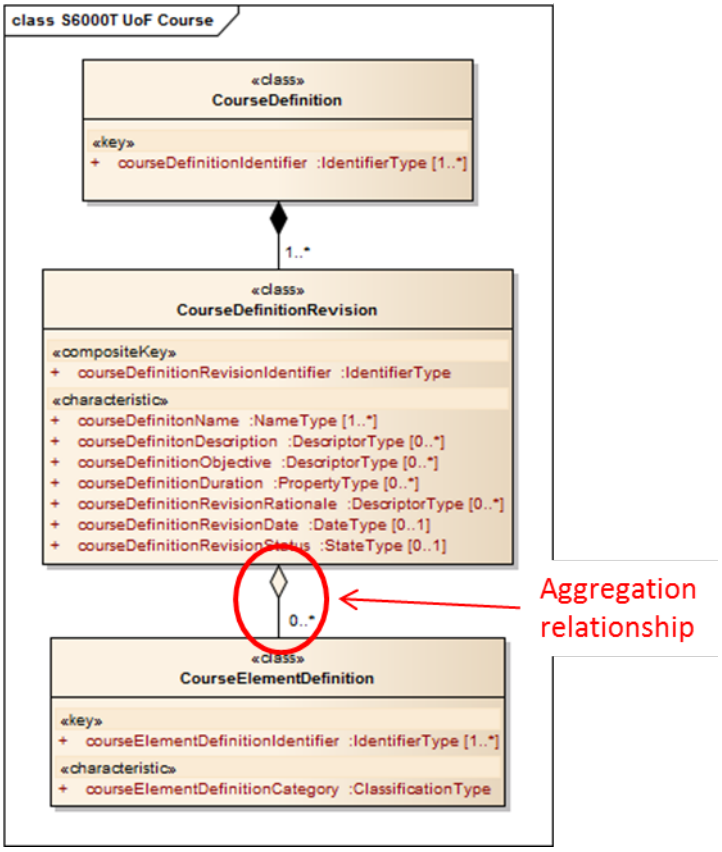
2.7 Aggregation

Aggregation defines the relationship of different Class instances, where one Class instance is a part of another Class instance. The individual classes exist independently and therefore part instances can outlive the parent class instances.

An example of an aggregation relationship is car and its wheels, where the car represents the whole class and the wheels represents parts of the complete car. The wheel class instances can exist independently of the car class instance.

Another example of an aggregation relationship is course and course elements. Refer to Fig 10. A course element can represent eg a lesson in safety measures which is included in many different courses.

Aggregations are illustrated by an arrow from the part class to the parent class using an empty diamond shape on the end.



ICN-B6865-SX004G0018-001-01

Fig 10 Example of aggregation relationship between classes in UML

An example of a relational representation for the aggregation association example in Fig 10 is illustrated in Fig 11. The example represents this association as a table that establishes the relationship between a course and its course elements using the key values for the respective Class.

Note

The Class `CourseAndCourseElementRelationship` is not directly visible as a Class in the UML Model in Fig 10, because it is the representation of the aggregation between the `CourseDefintionRevision` Class and the `CourseElementDefintion` Class.

CourseAndCourseElementRelationship		
courseId	courseRevId	courseElementId
C-234	A	LSN-34
C-234	A	LSN-56
C-123	A	LSN-34

ICN-B6865-SX004G0019-001-01

Fig 11 Example on relational representation for a aggregation relationship

2.8 Composition aggregation

Composition aggregation is another form of whole and part relationship, where the children are dependent upon the lifecycle for the parent Class (ie, instances of the child Class cannot exist without its parent Class instance).

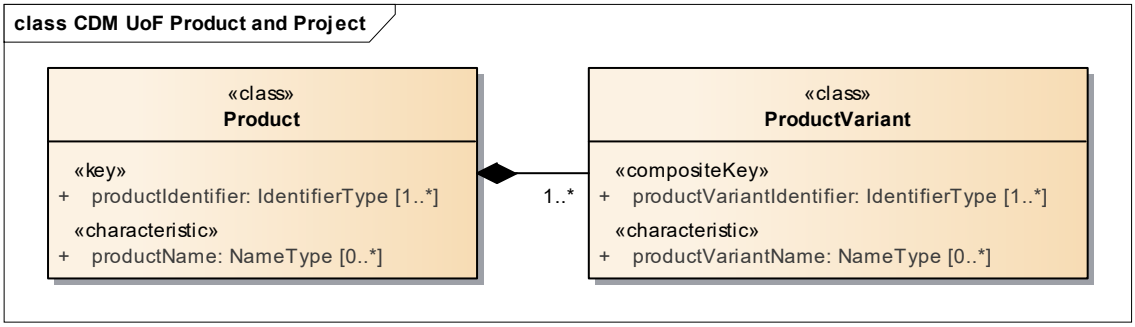
Composition aggregations are illustrated by an arrow with a filled diamond shaped head pointing from the child Class to the parent Class.

Note

The same arrow style is used to connect attribute groups to their classes, but has a slightly different meaning (Refer to Para 2.4).

An example of a composition relationship is the possibility to create various variants of a Product. A variant of a Product only make sense if a Product exists. A composition Class must include at least one attribute defined as <<compositionKey>>. A <<compositionKey>> attribute indicates that the key defined for the composition Class is not enough to uniquely identify an instance of that Class, but it must be combined with the key for its parent Class.

Fig 12 illustrates a composition relationship between the Product Class and the ProductVariant Class. This relationship expresses that a ProductVariant cannot be defined if it is not defined in the context of a parent Product, and it cannot be uniquely referred to if not both the productIdentifier and the productVariantIdentifier is given.



ICN-B6865-SX004G0011-002-01

Fig 12 Example of a composition aggregation in UML

Composition relationships can be viewed as two relational tables, where the relational table that represents the child class has a primary key which constitutes of both the key attributes for the whole (composition) class, plus an additional key attribute that distinguishes instances of the part (child) class. Refer to Fig 13.

Product	
productIdentifier	**productName**
1B	Eurofighter

ProductVariant		
productIdentifier	**productVariantIdentifer**	**productVariantName**
1B	GS	German Single Seat
1B	GT	German Twin Seat

ICN-B6865-SX004G0012-001-01

Fig 13 Example of possible relational representation of the composition aggregation given above

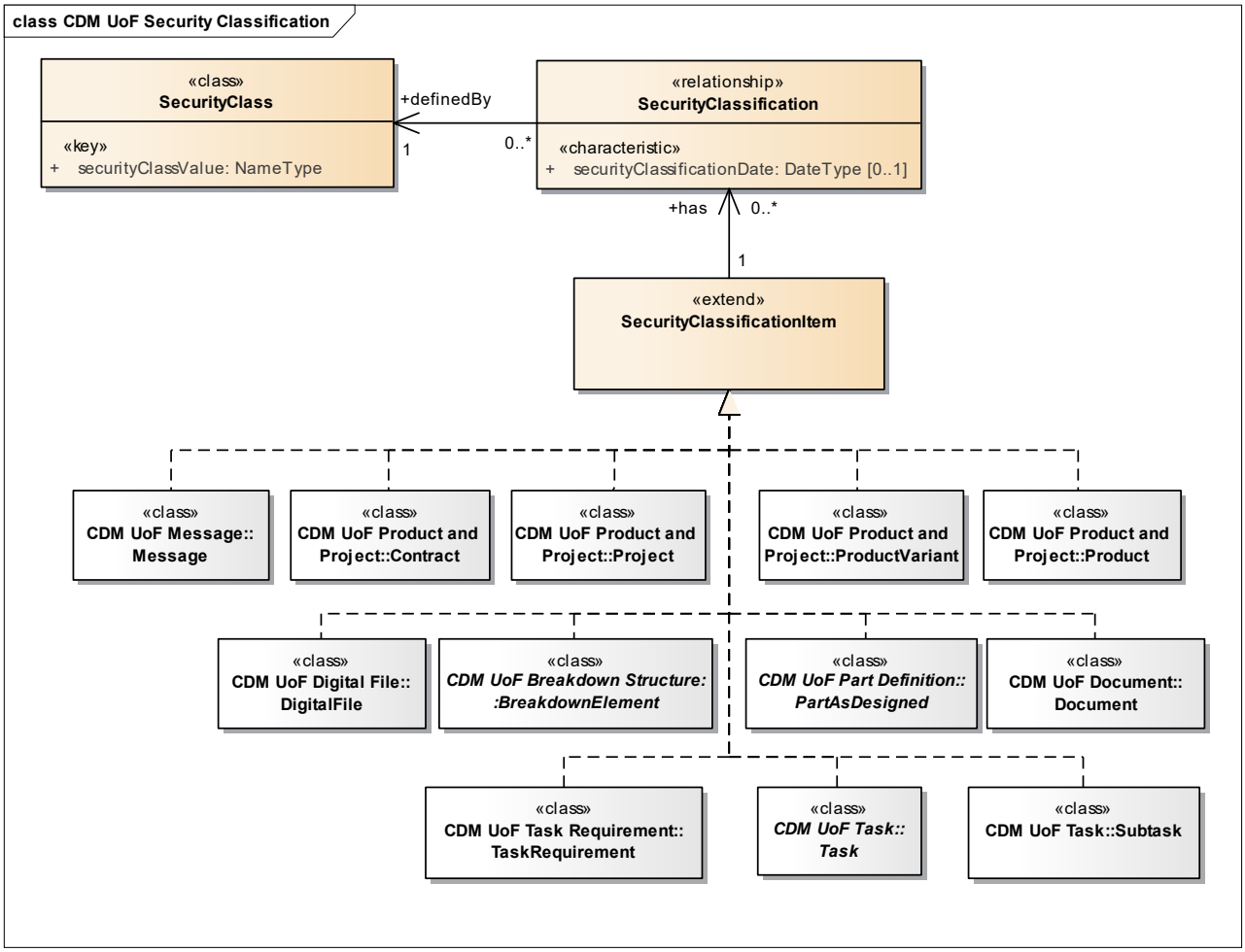
2.9 Extension

An <<extend>> is a way of modelling features that are common for a set of classes.

This means that any Class that realizes (implements) an <<extend>> must also support all features defined by the <<extend>>. These features include its attributes, relationships and behavior.

An example of an <<extend>> is SecurityClassificationItem. This <<extend>> is realized by Product, ProductVariant, BreakdownElement, PartAsDesigned, etc. (Refer to Fig 14). This means that the attributes and relationships that are defined for the SecurityClassificationItem <<extend>> must be supported (implemented) by all classes that realizes the <<extend>>. Instances of all those classes will have the capability to have one or many assigned security classifications (SecurityClass).

The graphical representation for the realize relationship (<<extend>>) is almost identical to the generalization relationship, except that it is a dashed line instead of a solid line.



ICN-B6865-SX004G0013-002-01

Fig 14 Example of and <<extend>>

The <<extend>> definition can be viewed as adding columns to existing relational tables as well as adding relationship tables that will represent the extend associations. Refer to Fig 15.

SecurityClass
securityClassValue
NATO RESTRICTED
NATO SECRET
NATO TOP SECRET

Product		
productIdentifier	**productName**	**securityClassValue**
1B	Eurofighter	NATO RESTRICTED

ProductVariant		
productIdentifier	**productVariantIdentifer**	**securityClassValue**
1B	GS	NATO RESTRICTED
1B	GT	NATO RESTRICTED

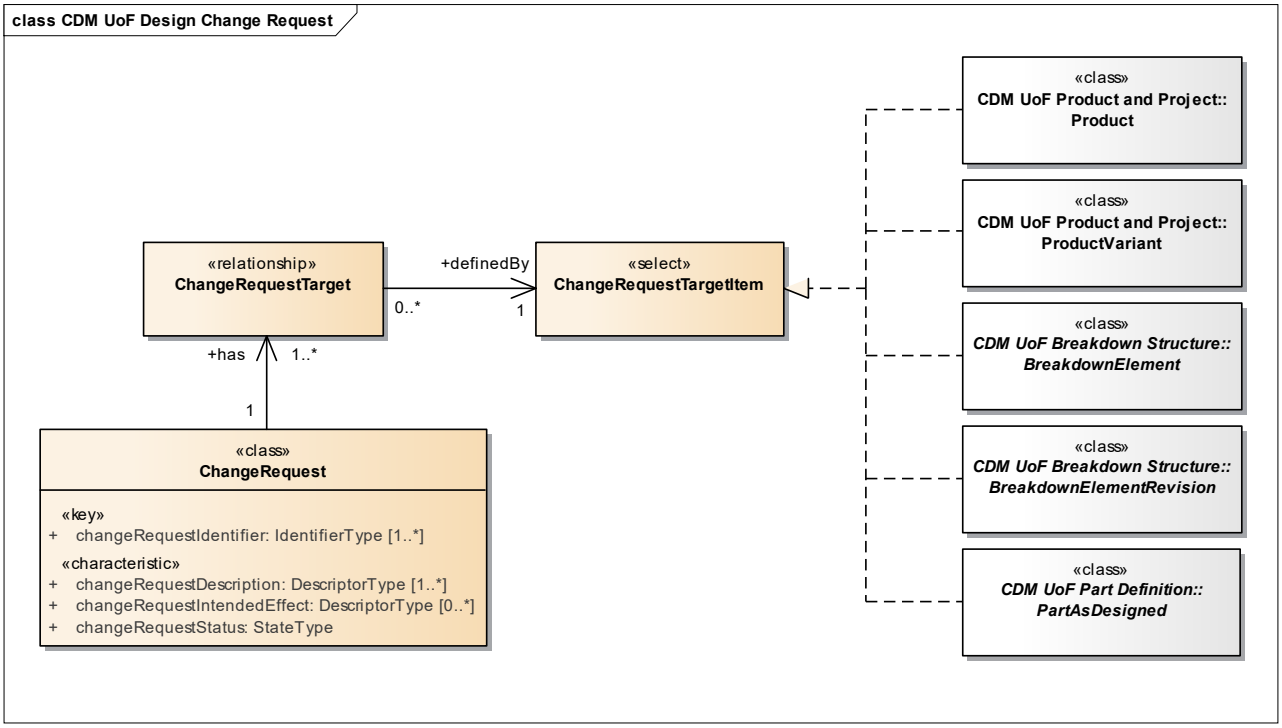
ICN-B6865-SX004G0014-002-01

Fig 15 Example of possible relational representation of <<extend>> interface

2.10 Selection

Within the S-Series IPS specifications UML model, `<<select>>` is used to express choices between different classes. This represents a "One of" relationship.

Fig 16, shows that a `ChangeRequest` can have multiple `ChangeRequestTarget`s each being defined by a `ChangeReqestTargetItem`, which can be one of `Product` or `ProductVariant` or `BreakdownElement`, etc.



ICN-B6865-SX004G0015-002-01

Fig 16 Example of a <<select>> to illustrate choices

3 Additional guidance for reading the S-Series IPS specifications UML class models

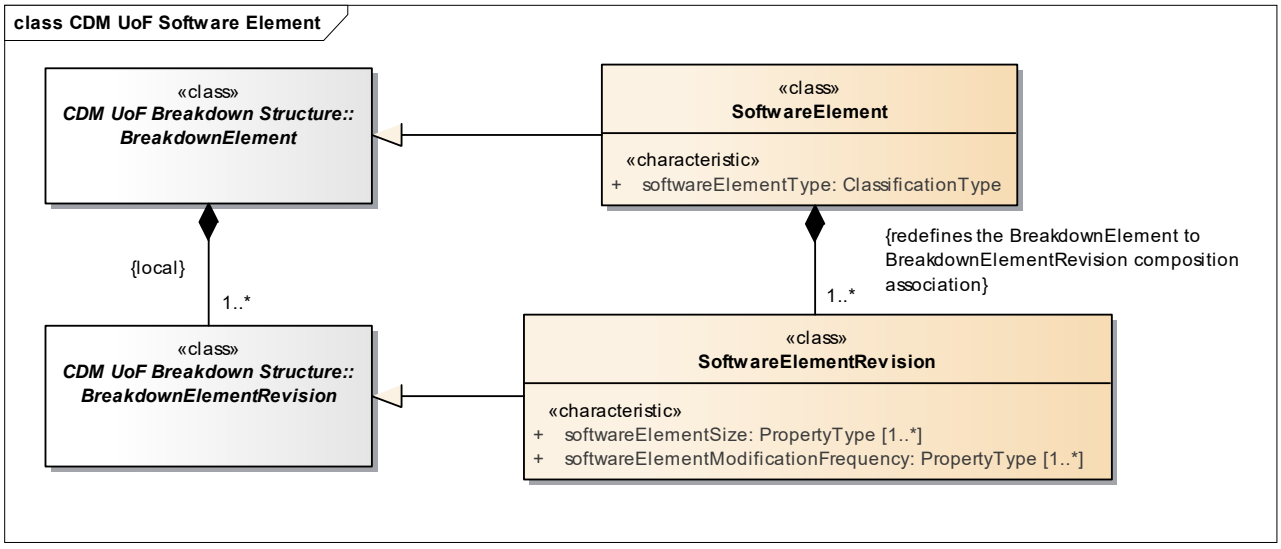
3.1 Class coloring

The class models contain classes that are both filled and not filled (white). A filled class represents classes that are introduced and defined within the Unit of Functionality (UoF) under consideration. Classes that are not filled are classes that are defined in other UoFs. Classes that are defined in other UoFs are always presented without any of their attributes except if there are any additional attributes which are added to the Class in the context of the UoF.

Note

Each class is only defined in one UoF in a particular S-Series IPS Specifications UML class model. (ie, it is only filled once).

Fig 17 illustrates that `SoftwareElement` and `SoftwareElementRevision` are fully defined in the UoF Software Element. However, `BreakdownElement` and `BreakdownElementRevision` are defined in another UoF namely UoF Breakdown Structure.



ICN-B6865-SX004G0016-002-01

Fig 17 Example of a Class defined in another UoF

3.2 Local associations

Local associations are a specific construct to indicate a consistency restriction in the context of generalized classes. A composition aggregation between Classes annotated with the `{local}` label (refer to Fig 17) indicates, that if there is a specialization on the parent Class the corresponding specialization of the child Class has to be used.

Local associations are only applicable to generalized classes.

The examples shown in Fig 17 defines that there is always a composition relationship between a `BreakdownElement` and its `BreakdownElementRevision`s but that each specialization of `BreakdownElement` (eg, `SoftwareElement`) will redefine the type of class that must be used as its revision Class (eg, `SoftwareElementRevision` in the context of `SoftwareElement`.

4 S-Series attribute data types

4.1 Overall description

The data types used to define Class attributes in the S-Series IPS specifications are not the same as those typically used in other data models (eg, integer, real, and string). In order to create a richer data model, a set of S-Series IPS specifications data types (primitives and compound attributes) are used. These primitives and compound attributes have been defined by the DMEWG.

Note

The data types used in the S-Series IPS specifications data models are based on the basic ISO 10303:239 Product Life Cycle Support (PLCS) templates, and the OASIS usage guidance on how to implement PLCS.

For example, consider part identifier (part number). In a traditional data model, a part identifier would typically be represented as a string value. There is often, for example, no additional information about the organization that provided the part identifier.

However, in the S-Series IPS specifications all identifiers are always defined to be of the primitive data type `IdentifierType`. In this way, the string value representing the part number can always be defined together with the following attributes:

- Information about the type of part identifier that is being represented
- Identification of the organization that created ("owns") the part identifier
- Information on how the organization that assigned the part identifier is identified (eg, CAGE code).

There are three groups of attribute types that are used throughout the S-Series IPS specification's data models. These are categorized as `<<primitive>>`, `<<umlPrimitive>>` and `<<compoundAttribute>>` respectively.

4.2 <<primitive>> data types

The primary purpose for the S-Series IPS specifications `<<primitive>>` data types is to establish a set of predefined patterns that allow for further characterizations of attribute values in order to record valuable metadata for each value. These characterizations are defined to support eg, recording multiple values over time, keeping values in different languages, etc. This also enables applicability statements to be part of the additional characterization that can be associated with single attribute values (eg, define that a given value is only applicable to arctic conditions).

The following `<<primitive>>` data types are used by the S-Series IPS specifications:

- `IdentifierType`
- `ClassificationType`
- `DateType`
- `DateTimeType`
- `DescriptorType`
- `NameType`
- `PropertyType`
- `StateType`

4.2.1 IdentifierType

The `IdentifierType` `<<primitive>>` is used to represent any kind of identification. Any attribute that is defined to be of data type `IdentifierType` holds the value for the identifier string but it does also include the possibility to record additional optional

characterizations such as a classifier that determines the type of identifier that is provided along with an identification of the organization that owns/sets the identification.

Note
The optional characterizations allow for multiple values to be defined for an attribute with a cardinality greater than one and still be able to distinguish between them.

4.2.2 ClassificationType

The `ClassificationType` `<<primitive>>` is used to represent attributes where a finite set of values can be used to characterize the associated object for a defined purpose.

Note
In legacy applications those often appear as value lists (enumerations).

4.2.3 DateType

The `DateType` `<<primitive>>` is used to represent Gregorian calendar dates.

4.2.4 Date Time Type

The `DateTimeType` `<<primitive>>` is used to represent Gregorian calendar date with time down to seconds using the 24-hour time scale. It can also include an oriented offset from the Coordinated Universal Time.

4.2.5 DescriptorType

The `DescriptorType` `<<primitive>>` is used to represent any form of textual data (free form). Any attribute that is defined to be of data type `DescriptorType` holds the value for the text string but it does also include the possibility to record the language in which the text is provided, the organization that provided the text and the date when it was provided.

Note
The optional characterizations allow for multiple values to be defined for an attribute with a cardinality greater than one and still be able to distinguish between them.

4.2.6 NameType

The `NameType` `<<primitive>>` is used to represent informal identifications. Any attribute that is defined to be of data type `NameType` holds the value for the name text string, but it does also include the possibility to record the language in which the name is given and the organization that assigned the name.

A `NameType` is used to store an informal identification for a class whereas the `DescriptorType` represents more narrative text. The `NameType` attribute can be sufficient for a Human to identify a specific Class instance, while might not be a real unique identifier for the object.

To illustrate the difference between `IdentifierType`, `NameType` and `DescriptorType` consider the example of the company Saab. "`Saab`" itself would be a `NameType` attribute, because it is not possible to identify a specific company by that. An `IdentifierType` attribute could be populated with the CAGE code "`S8372`" while a `DescriptorType` attribute could be populated with a lengthier description of the company goal.

Note
The optional characterizations allow for multiple values to be defined for an attribute with a cardinality greater than one and still be able to distinguish between them.

4.2.7 PropertyType

The PropertyType <<primitive>> is used to represent measurable characteristics. Any attribute that is defined to be of data type PropertyType holds the value together with the unit of measure, but it also includes the possibility to record when the property value was recorded as well as information on the method by which the value has been determined (eg, estimated, or calculated).

Note:
The optional characterizations allow for multiple values to be defined for an attribute with a cardinality greater than one and still be able to distinguish between them.

There are four specializations of PropertyType which can be used to represent different types of property values. These are:

- SingleValuePropertyType, which is a PropertyType that specifies a single value and its unit
- ValueWithTolerancesPropertyType, which is a PropertyType that specifies a range of values by specifying a single nominal value and its unit together with the permitted variations from the nominal value
- ValueRangePropertyType, which is a PropertyType that specifies a value pair that represents the range limits and their unit
- TextPropertyType, which is a PropertyType that specifies the value in text format

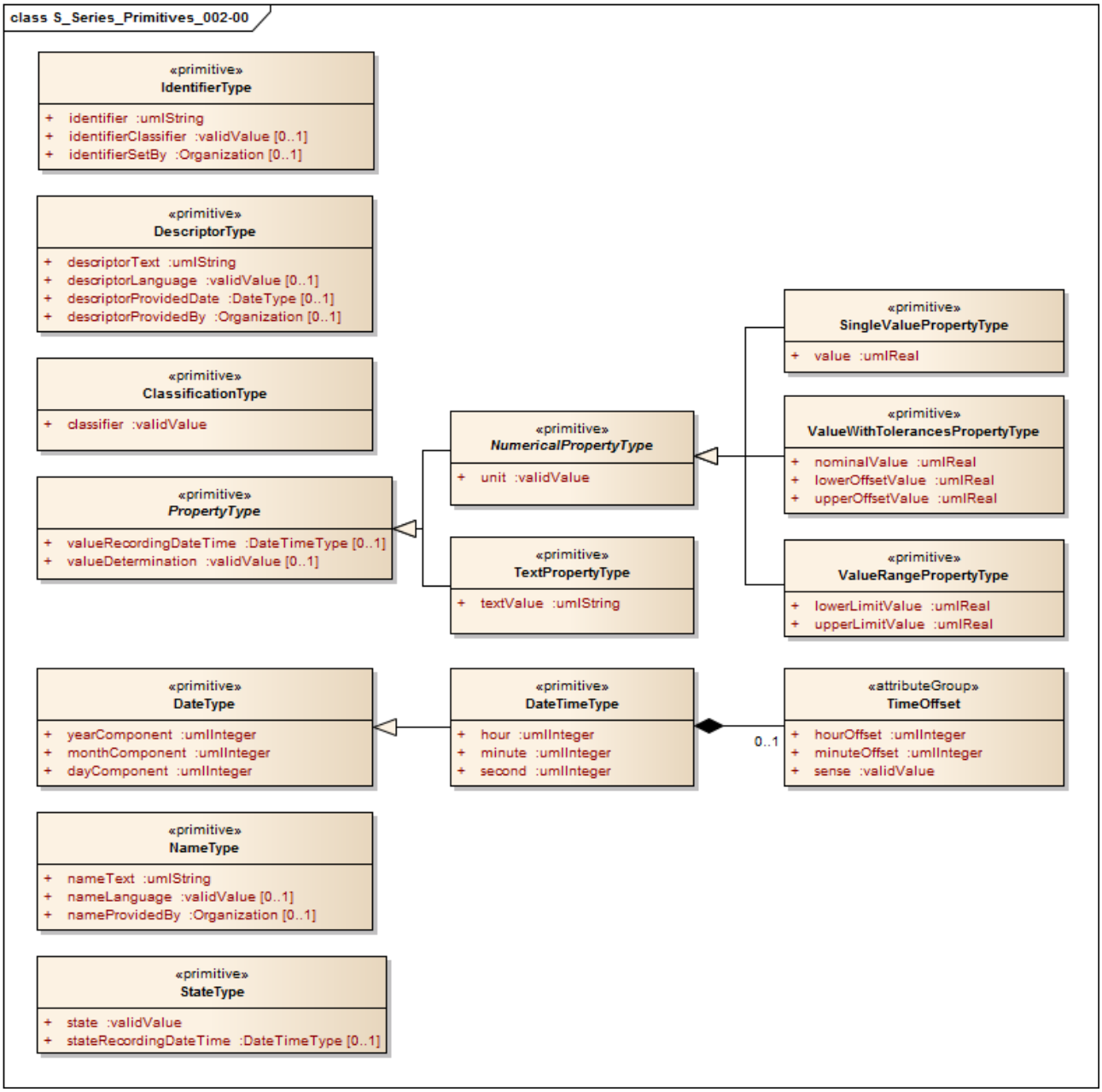
4.2.8 StateType

The StateType <<primitive>> is used to represent a particular condition that applies to a Class instance (object) at a particular point in time. Any attribute that is defined to be of data type StateType holds the state itself but it also includes the possibility to record the calendar date and time when the state was recorded.

Note:
The optional characterizations allow for multiple values to be defined for an attribute with a cardinality greater than one and still be able to distinguish between them.

4.2.9 Graphic Representation

Fig 18 illustrates the <<primitive>> data types described here.



ICN-B6865-SX004G0020-001-01

Fig 18 <<primitive>> data types

4.3 <<umlPrimitive>> data types

UML primitives are S-Series IPS specification's UML representations of values where there is no need for any additional metadata characterizations with respect to the value itself. The rationale for defining S-Series IPS specifications specific `<<umlPrimitive>>` representations for the respective UML primitive, is to enable the possibility to associate any given value with (eg, a remark, a document reference, etc). but also enable its use in project specific attributes.

The following UML primitives have a corresponding `<<umlPrimitive>>`:

– Boolean

- Integer
- String
- Real
- Unlimited natural

4.3.1 umlBoolean

The umlBoolean <<umlPrimitive>> is used to represent the Boolean values "true" and "false".

4.3.2 umlInteger

The umlInteger <<umlPrimitive>> is used to represent integer values such as "1", "2", etc.

4.3.3 umlReal

The umlReal <<umlPrimitive>> is used to represent the mathematical concept of real numbers.

4.3.4 umlString

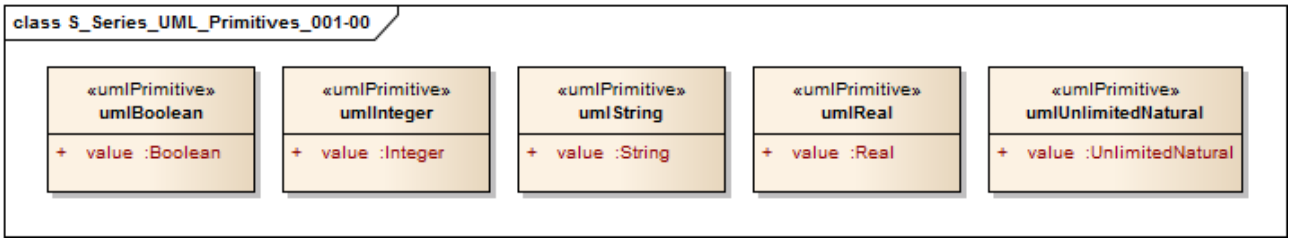
The umlString <<umlPrimitive>> is used to represent the sequence of characters. Character sets can include non-Roman alphabets and characters.

4.3.5 umlUnlimitedNatural

The umlUnlimitedNatural <<umlPrimitive>> is used to represent unlimited natural values.

4.3.6 Graphic Representation

Fig 19 illustrates the <<umlPrimitive>> data types described here.



ICN-B6865-SX004G0021-001-01

Fig 19 <<umlPrimitive>> data types

4.4 <<compoundAttribute>> data types

The primary purpose for the S-Series IPS specifications <<compoundAttribute>> data type is to define often recurring patterns of attributes that are used together. The following <<compoundAttribute>> data types are used by the S-Series IPS specifications:

- SerialNumberRange
- DatedClassification
- TimeStampedClassification
- DateRange
- DateTimeRange
- AuthorizedLife
- ThreeDimensional and its specializations:
  - Cuboid

- Cylinder
- Sphere

4.4.1 SerialNumberRange

The SerialNumberRange <<compoundAttribute>> is used to identify a possibly open-ended interval of serialized items.

4.4.2 DatedClassification

The DatedClassification <<compoundAttribute>> is used to represent classifications where the date when the classification was done is as important as the classification itself.

4.4.3 TimeStampedClassification

The TimeStampedClassificaiton <<compoundAttribute>> is a combination of a classification where the time stamp (consisting of date and time) for when the classification was done is as important as the classification itself.

4.4.4 DateRange

The DateRange <<compoundAttribute>> identifies an interval of dates.

4.4.5 DateTimeRange

The DateTimeRange <<compoundAttribute>> identifies an interval of date and times.

4.4.6 AuthorizedLife

The AuthorizedLife <<compoundAttribute>> is used to represent authorized life limits together with its authorizing organization.

4.4.7 ThreeDimensional

The ThreeDimensional <<compoundAttribute>> represents spatial magnitudes.

4.4.7.1 Cuboid

The Cuboid <<compoundAttribute>> represents a three-dimensional object where all its faces are rectangles and all angles are right angles.

4.4.7.2 Cylinder

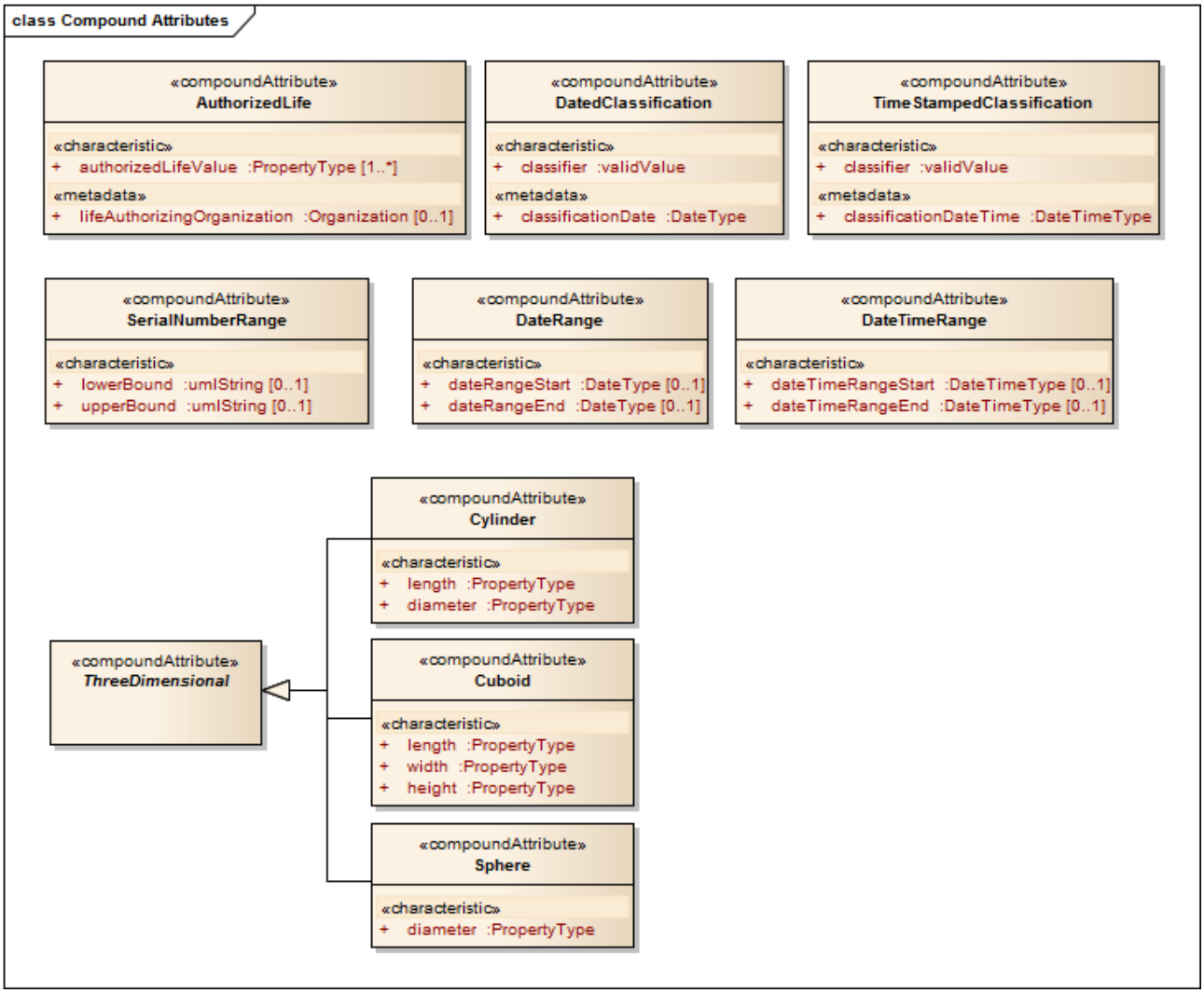
The Cylinder <<compoundAttribute>> represents a three-dimensional object with straight parallel sides and a circular section.

4.4.7.3 Sphere

The Sphere <<compoundAttribute>> represents a three-dimensional object where every point on its surface is equidistant from its center.

4.4.8 Graphic Representation

Fig 20 illustrates the <<compoundAttribute>> data types described here.



ICN-B6865-SX004G0022-001-01

Fig 20 <<compoundAttribute>> data types

4.5 Organization as the data type

In some cases, `Organization` is used as the data type for an attribute. Where this occurs, it must be read as that there is a reference to an `Organization` and the reason for the reference is determined by the attribute name (eg, `authorizingOrganization` for an authorized life attribute).

4.6 Valid value as the data type

Valid value as the data type indicates that there is a finite set of values, which can be used as the value for the defined attribute. These allowed values must also be defined as part of the XML schemas that controls an implemented data exchange.