Unified Modeling Language (UML) model readers' guidance

SX004G-B6865-0X001-00

Issue No. 1.0

Publishers:

AeroSpace and Defence Industries Association of Europe

Aerospace Industries Association

Table of contents

The listed documents are included in Issue 1.0 dated 2016-08-31, of this publication.

Copyright and user agreement

1 Copyright

All correspondence and queries should be directed to:

ASD
10 Rue Montoyer
B-1000 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 model reader's guidance SX004G
- examples (eg, XML instances, UML models, pdf files, style sheets) and schemas
- any other software or information under the heading "**SX004G™ suite of information**", available for download from 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 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.

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 guide for the use of the S-Series Integrated Logistics Support (ILS) specifications
SX002D	Common data model for the S-Series ILS specifications
S2000M	International specification for material management – Integrated data processing for military equipment
S3000L	International procedure specification for Logistic Support Analysis (LSA)
http://www.uml.org	Unified Modelling Language (UML)

1 General

The UML model reader's' guidance for the S-Series Integrated Logistic Support (ILS) specifications is a document describing how to read and understand the Unified Modelling Language (UML) class models that are created for the S-Series ILS specifications (eg, S2000M, S3000L, etc), including the common data model. Refer to SX002D.

2 Purpose

The purpose of the UML model reader's guidance is to provide clear instruction on how the S-Series Specifications´ 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 ILS. The work was 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 (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 integrated logistics support specifications and jointly develop the S-Series ILS specifications. This MOU authorized the formation of the AIA/ASD ILS Specifications Council, whose responsibilities include performing liaison between ASD and AIA, developing and maintaining the S-Series ILS 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 ILS specifications. Its creation and maintenance was 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 the CDM to understand and read it´s UML constructs, the need for an UML readers' guidance was recognized. It is numbered SX004G and created, maintained and assigned towards 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 - Human performance analysis, training analysis and training design
SX000i	International guide for the use of the S-Series Integrated Logistic Support (ILS) specifications
SX002D	Common data model for the S-Series ILS specifications
SX003X	Compatibility matrix for the S-Series ILS Specifications
SX004G	UML model readers' guidance

1 General

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

2 Scope

Within the scope of the document are:

- Illustrations of UML constructs used within the S-Series ILS specifications
- Explanation of how to read and interpret the constructs

Note

The rules for the assembly and creation of a valid UML class model for the S-Series specifications is out of the scope of this guide

3 S-Series ILS specifications

Multiple AeroSpace and Defense Industries Association of Europe (ASD) and Aerospace Industries Association of America (AIA) ILS 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 - Human performance analysis, training analysis and training design
- SX000i - International guide for the use of the S-Series Integrated Logistic Support (ILS) specifications
- SX002D - Common data model for the S-Series ILS specifications
- SX003X - Compatibility matrix for the S-Series ILS specifications
- SX004G - UML model readers' guidance

Chapter 1.3

How to use the specification

Table of contents

Page

List of tables

References

Table 1 References

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

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 readers´ guide

Chap 2 includes the UML model readers' 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.

Chapter 1.4

Maintenance of the specification

Table of contents

List of tables

References

Table 1 References

Chap No./Document No.	Title
SX000i	International guide for the use of the S-Series Integrated Logistic Support (ILS) 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 Logistic Support (ILS) specifications Council. Refer to SX000i.

Both the DMEWG and the ILS specifications Council include representatives from AeroSpace and Defense Industries Association of Europe (ASD) and Aerospace Industries Association (AIA) member companies and nations.

Chapter 2

UML model reader´s guidance

Table of contents

List of tables

References

Table 1 References

Chap No./Document No.	Title
None	

1 General guidance

The Unified Modeling Language™ (UML) 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.

The S-Series ILS specifications use only the UML class model, which defines a static view of the information (classes, attributes and relationships) that are needed to support the business processes.

Class models are the most widely used part of UML. They show the things that are to be represented, and their relationships.

This chapter gives an overview of the UML constructs used in the S-Series ILS specifications data models. The data models conform to the UML Writing Rules and Style Guide published as an internal document to modelers by the AeroSpace and Defense Industries Association of Europe (ASD) and Aerospace Industries Association (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 read the S-Series ILS specifications data models.

Some UML class model concepts are also translated into a relational database example. These examples are provided for those readers that have an understanding of relational databases, but 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. They should not be seen as the solution for an implementation.

1.1 Class

The basic concept within a class diagram is a box called “class”. The classifier gives the name of the class together with an enumeration of its attributes.

Note

Within the S-Series ILS specifications UML models the possibility to describe “methods” is not used

A class can have one or many attributes. Each attribute is presented with its attribute name, classification, data type, visibility and cardinality.

Attributes are divided into groups.

Each group is shown within double angle brackets (<<…>>) above the attribute name.

Attributes that are part of the key (primary key) are classified as <<key>> attributes. Attributes that comprise the key are always defined first in the attribute list for the Class.

Attributes that define the characteristics of a given object (class instance) are classified as <<characteristic>>. Characteristics typically include measurable properties, classifications and descriptions.

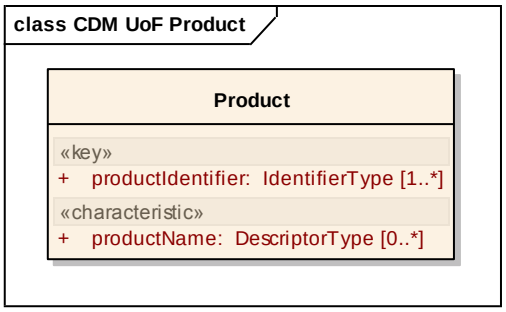
Metadata for a class instance (object) provide information about one or more aspects of the class instance, such as the means of creation, author, time and date of creation etc. Metadata attributes are classified as <<metadata>>.

There are many cases where there is a need to provide additional information (metadata/characterization) for a given attribute value, eg, the date when a classification was

done, or the time when a property value was measured. These attributes are classified as <<characteristicMetadata>>. Characteristic metadata are always shown directly after the attribute to which it applies.

The data type and cardinality for an attribute is shown directly after the attribute name. Refer to Para 3. The cardinality for an attribute is defined in the same way as cardinality for an association. Refer to Para 1.4 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 a class is given at Fig 1. It shows the class Product with its two attributes, productIdentifier and productName. Each attribute also shows its classification, data type and cardinality.



ICN-B6865-SX004G0001-001-00

Fig 1 Example of a class in UML

A class can have zero, one or many instances. An example on a product instance is the New Fighter Aircraft Product.

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-SX004G0002-001-00

Fig 2 Example of relational representation of the UML class

Note

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

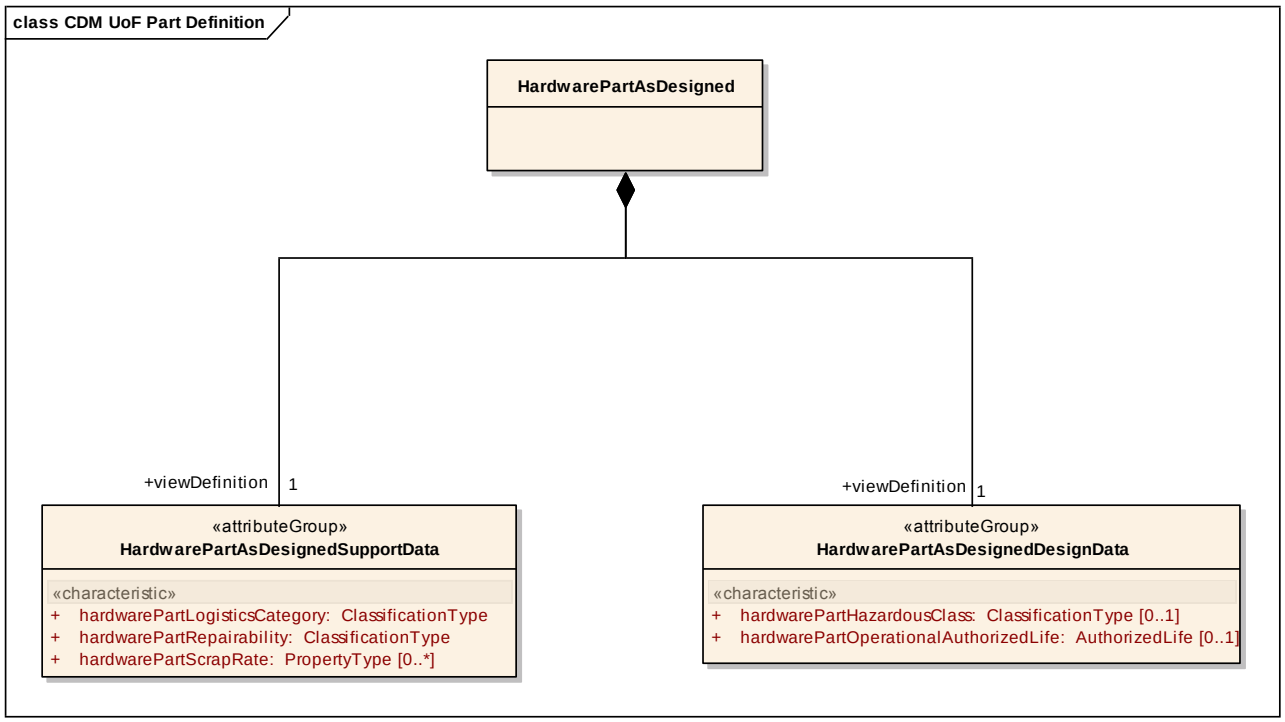
A class can also have relationships to other classes in terms of association, composition, aggregation, generalization/specialization and realization.

1.2 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>>. Note that these attributes are to be seen as any other attributes defined for the class it's associated with, and could as well have been defined within the parent class.

In the example in Fig 3, the attributes associated with the part as designed 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).



ICN-B6865-SX004G0003-001-00

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

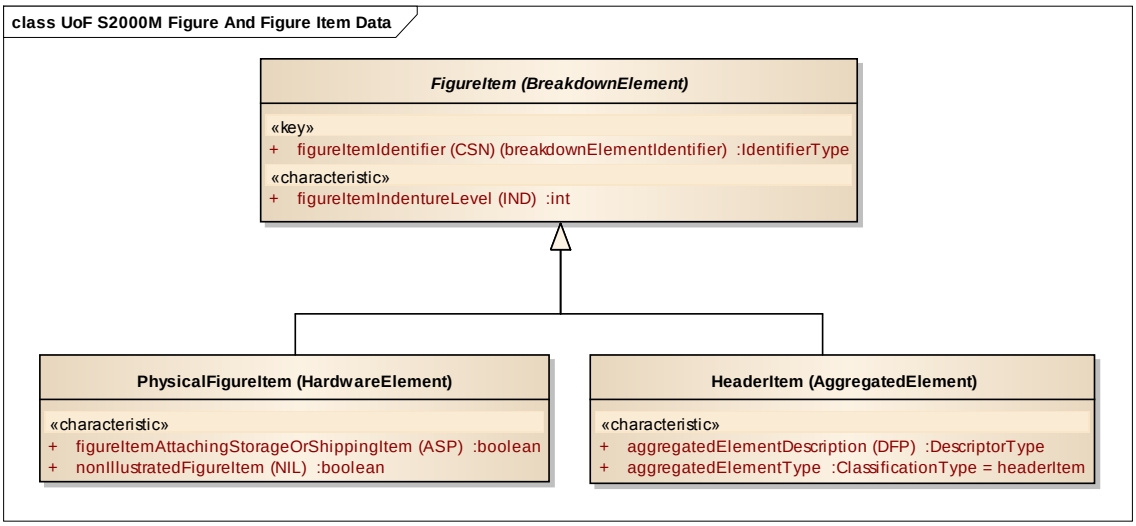
ICN-B6865-SX004G0004-001-00

Fig 4 Example of relational representation of an attribute group

1.3 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 specializations. Refer to Para 1.6.

In the example shown at Fig 5, FigureItem is an abstract class that cannot be instantiated directly. One either instantiates a PhysicalFigureItem or a HeaderItem. Either one of those inherit the attributes for figureItemIdentifier and figureItemIdentiferLevel from their abstract parent FigureItem.



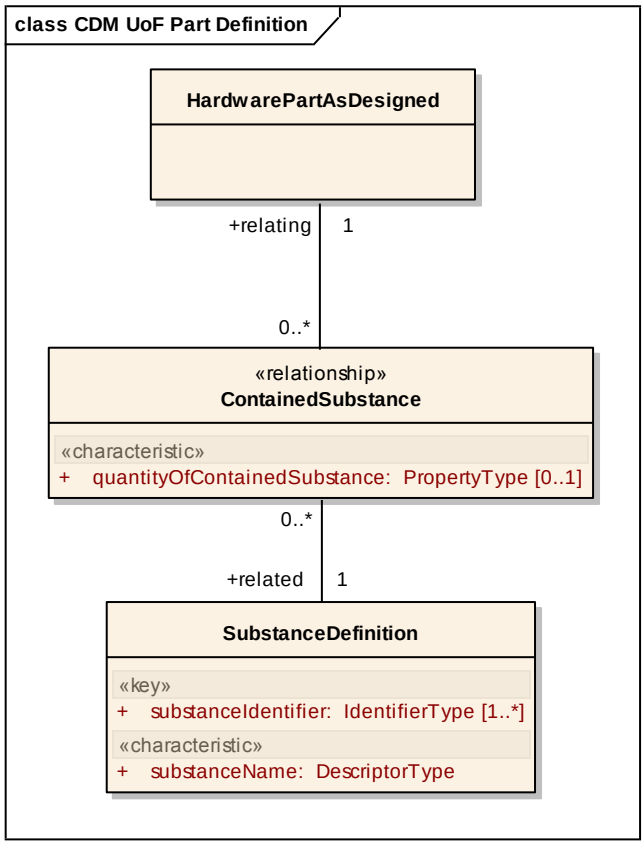
ICN-B6865-SX004G0005-001-00

Fig 5 Example of an abstract class

1.4 Association

Associations represent interdependencies between classes. Associations are often viewed as an attribute which is modelled as a class with a connector between the containing class and the class representing the attribute.

An example of an association is shown in Fig 6. It shows that a part (HardwarePartAsDesigned) can be associated with zero, one or many instances of contained substances. The substance itself (SubstanceDefinition) can be contained in zero, one or many parts. The cardinality for the association is given at the respective end of the association.



ICN-B6865-SX004G0006-001-00

Fig 6 Example of an association between classes in UML

Note

One-to-many and 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 example of an association (<<relationship>> class) that contains additional information (<<characteristics>>) about the association, is the ContainedSubstance <<relationship>> class. The ContainedSubstance <<relationship>> class has an attribute quantityOfContainedSubstance which determines the quantities for a given substance within a given part (eg, 50 gram).

Table 1 Association cardinalities used

Association	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..*	Zero, one or many instances of the associated class can be associated with each instance of the associating class (an optional association)

Association	Explanation
1..*	At least one instance of the associated class must be associated with each instance of the associating class (a mandatory association)

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 represents the primary keys from the respective associating and associated class. Table rows represent references to the associated instances.

Attributes that characterizes a <<relationship>> class can be viewed as additional columns in a relationship table in a relational database (quantityOfContainedSubstance).

The example uses the class diagram and shows how two parts (Engine and Radar) are connected with two substances (Quicksilver and Gold) that they contain. Furthermore an (optional) information on the quantity of each substance within the part is illustrated.

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

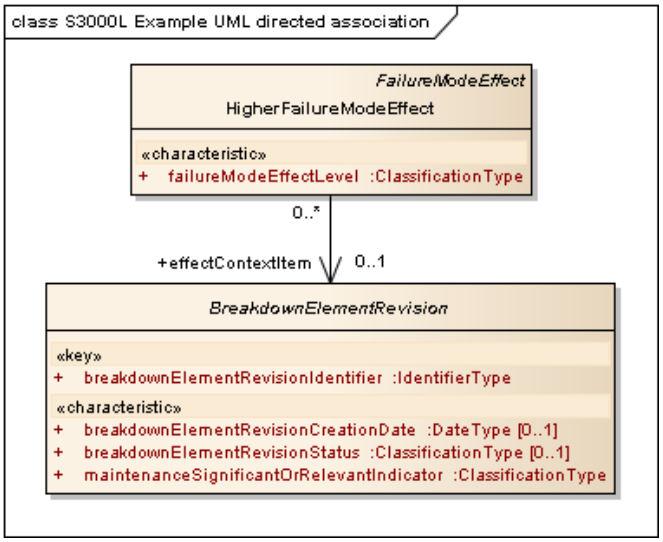
ICN-B6865-SX004G0007-001-00

Fig 7 Example of relational representation of an association with additional information

1.5 Directed association

A directed association has an open arrowhead at the target end of the association. In a directed association, two classes are related, but only one class knows that the relationship exists (the class with no arrow at the end).

An example of a directed association is the possibility to associate a higher failure mode effect with a breakdown element that represents the higher level function, including the end item. However, there is no requirement for the breakdown element to be able to navigate to any associated higher failure mode effects.



ICN-B6865-SX004G0008-001-00

Fig 8 Example of a directed association in UML

Note

A directed association in UML can be viewed as a relationship table in a relational database, in the same way as described for an association above.

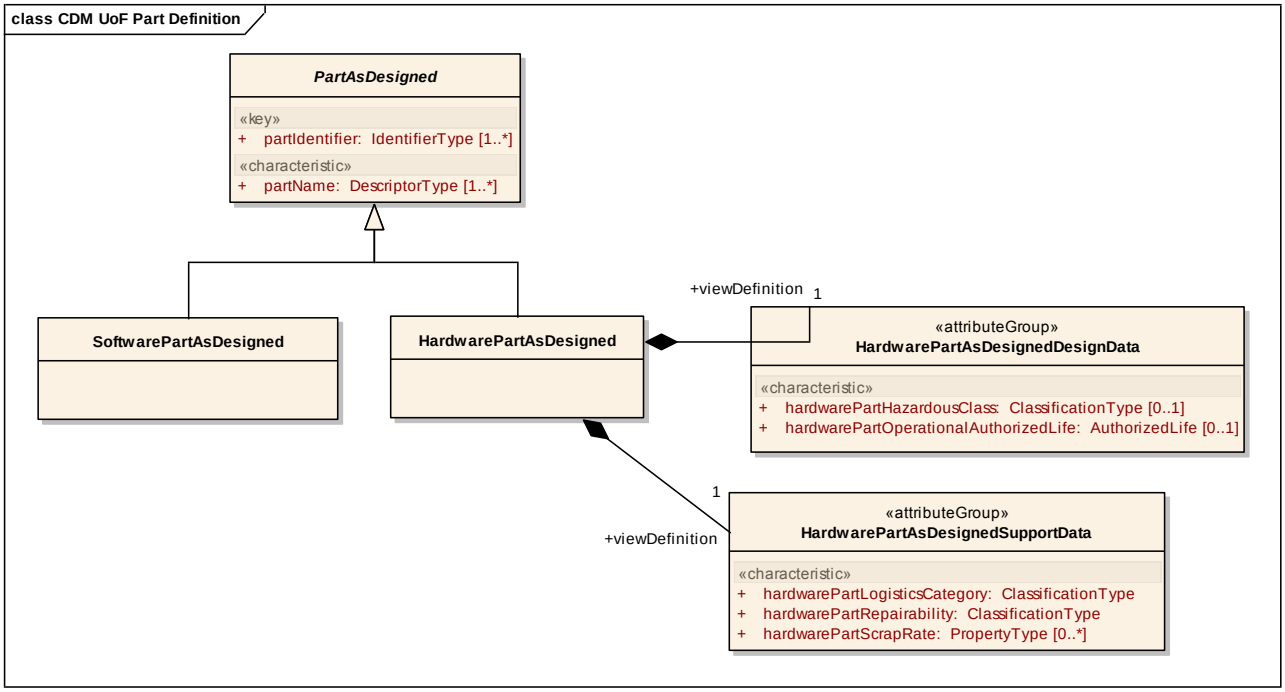
1.6 Generalization and specialization

Specialization refers to a 'is-a' relationship and is modelled as a solid line connector 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 of 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 part of specialization is substitutability. Substitutability means that wherever a parent class is being used, a parent can be replaced by a child.

An example of the specialization/generalization association is PartAsDesigned and its specialization into HardwarePartAsDesigned and SoftwarePartAsDesigned, respectively. In the example shown in Fig 9, the PartAsDesigned class has two attributes, partIdentifier and partName. This means that 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 could then also define additional attributes which are unique for its respective special characteristics (not in the example).



ICN-B6865-SX004G0009-001-00

Fig 9 Example of specialization (inheritance) relationships between classes in UML

Note

A special characteristic that can be defined for the parent class is that it can be defined as an 'abstract' class. This means that the parent class itself cannot be instantiated, but must be substituted by any of its child classes. This is shown in the model by making the class name (classifier) *italic*. PartAsDesigned in the example above is defined as an abstract class.

Specialization can be viewed as multiple tables with the same initial set of columns (incl. the primary key).

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

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

ICN-B6865-SX004G0010-001-00

Fig 10 Example of relational representation of the specializations given above

1.7 Aggregation

Aggregation defines whole and part relationships, ie, it indicates that one class is part of another class. In an aggregation relationship, the child class instances can outlive its parent class instance.

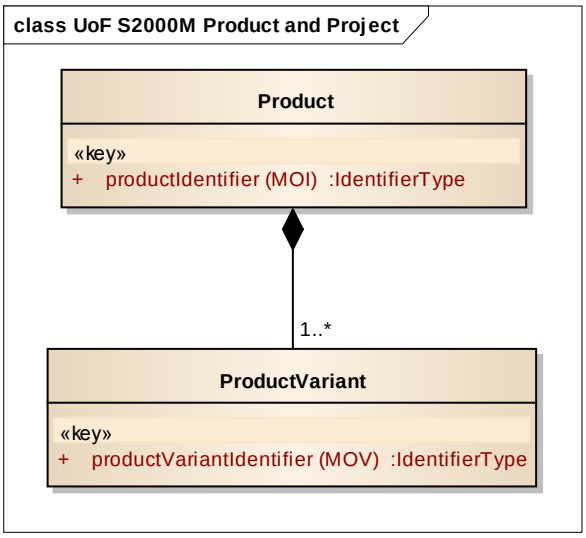
Note

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. However, the Wheel class instances can exist independently of the Car class instance.

1.8 Composition aggregation

Composition aggregation is another form of an aggregation relationship where the children are dependent upon the lifecycle for the parent class, ie, the child class instances cannot exist without its parent class instance.

An example of a composition aggregation is the possibility to create various variants of a product. A variant of a product does only make sense if a product exists.



ICN-B6865-SX004G0011-001-00

Fig 11 Example of a composition aggregation in UML

Composition aggregation 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 primary key for the whole (composition) class, plus an additional key that distinguishes instances of the child class.

Product	
productIdentifier	productName
1B	Eurofighter

ProductVariant		
productIdentifier	productVariantIdentifer	variantName
1B	GS	German Single Seat
1B	GT	German Twin Seat

ICN-B6865-SX004G0012-001-00

Fig 12 Example of relational representation of the composition aggregation given above

1.9 Interfaces

1.9.1 Interface to illustrate "common behavior"

Interfaces are a way of modelling features that are common for a set of classes.

Any class that realizes (implements) an interface must support the features defined by the interface. These features include its attributes, relationships and behavior.

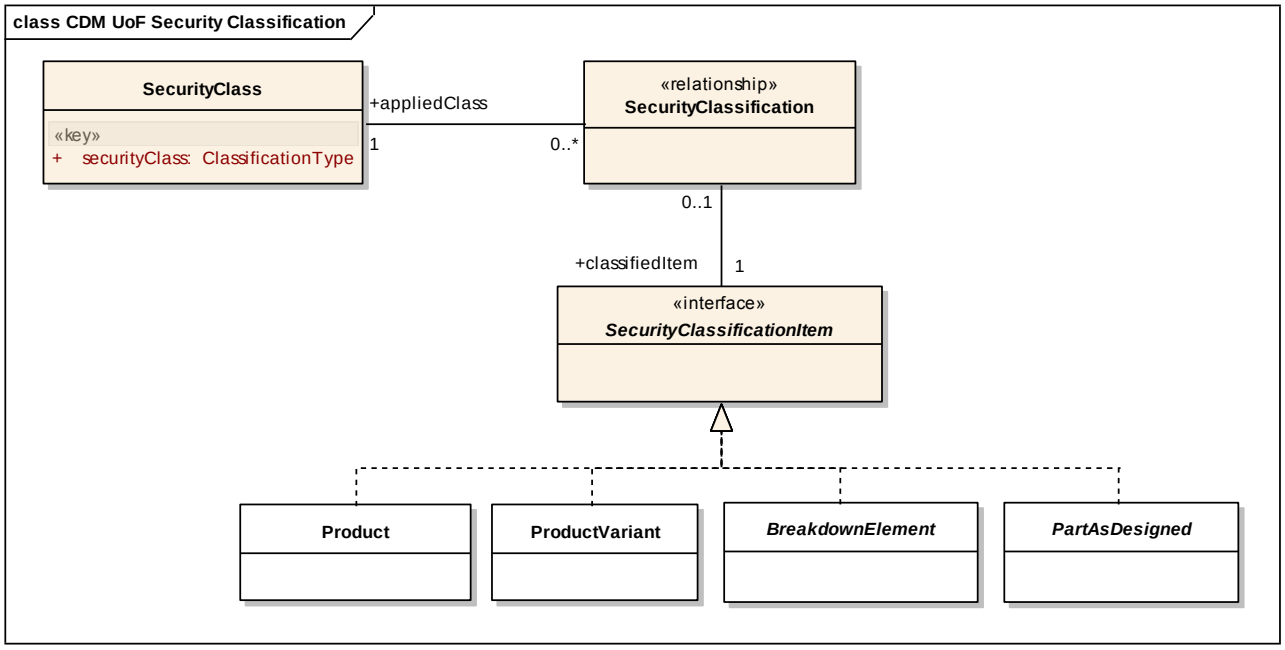
An example of an interface in the S-Series ILS Specifications data models is the SecurityClassificationItem. This interface is realized by Product, ProductVariant, BreakdownElement (Abstract Class) and PartAsDesigned (AbstractClass). This means that the attributes and relationships that are defined for the SecurityClassificationItem interface must be supported (implemented) by all four classes.

This means, that each of the four classes need to implement a way to relate none, one or more securityClass attributes to them

Note

Interfaces simplify the data model, eg, a similar relationship between one class and a set of other classes can be directed towards an interface which in turn is implemented by each class that can be part of that relationship.

The graphical representation for the realization connector is almost identical to the generalization connector, except that the connector is a dashed line instead of a solid line.



ICN-B6865-SX004G0013-001-00

Fig 13 Example of Interface and Realize relationships in UML

The Interface definition can be viewed as adding columns to existing relational tables (Refer to Fig 14 as well as adding relationship tables that will represent the interface associations.

SecurityClass	
securityClassID	**securityClassName**
1	NATO RESTRICTED
2	NATO SECRET
3	NATO TOP SECRET

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

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

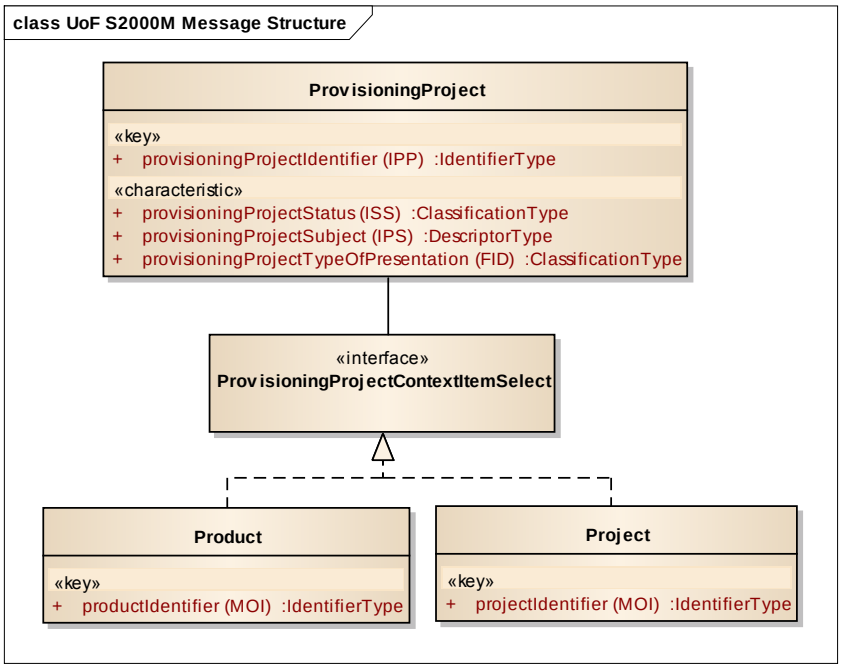
ICN-B6865-SX004G0014-001-00

Fig 14 Example on relational representation of interface definition

1.9.2 Interface to illustrate “choices”

Within the S-Series ILS specifications UML model, interfaces are also used to express choices between different classes. This represents an “either” “or” relationship.

Fig 15, shows that a ProvisioiningProject can either be related to a Product or to a Project.



ICN-B6865-SX004G0015-001-00

Fig 15 Example of an interface to illustrate choices

2 Special guidance for reading the S-Series ILS specifications UML class models

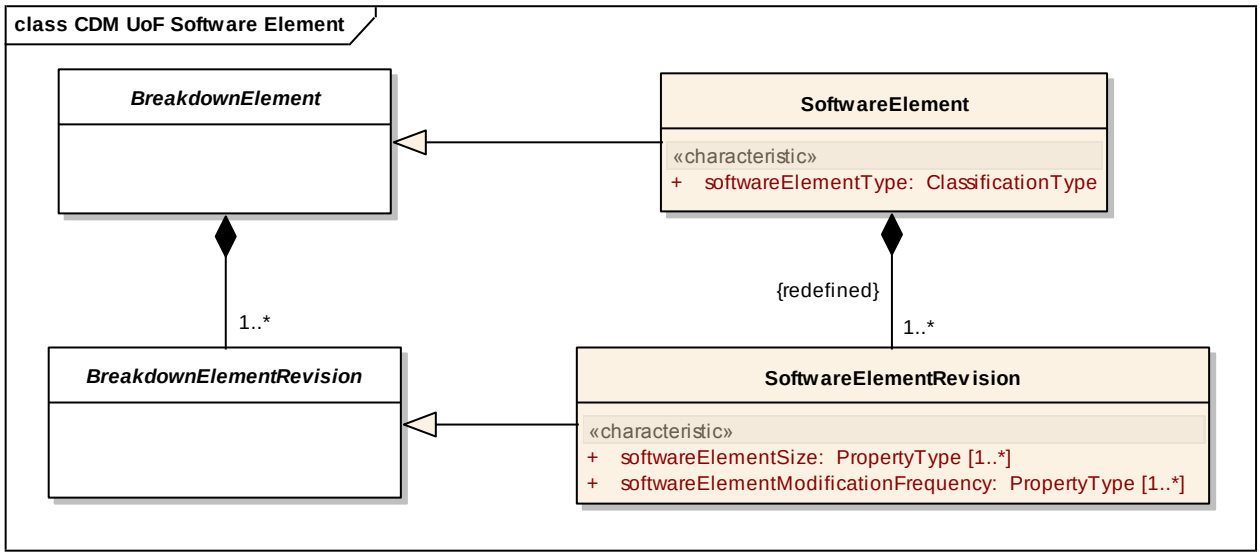
2.1 Classes and attributes

The data types used in the S-Series ILS specifications data models are based on the basic ISO 10303:239 Product Life Cycle Support (PLCS) constructs, and the OASIS usage guidance on how to implement PLCS. Refer to Para 3.

2.2 Diagrams

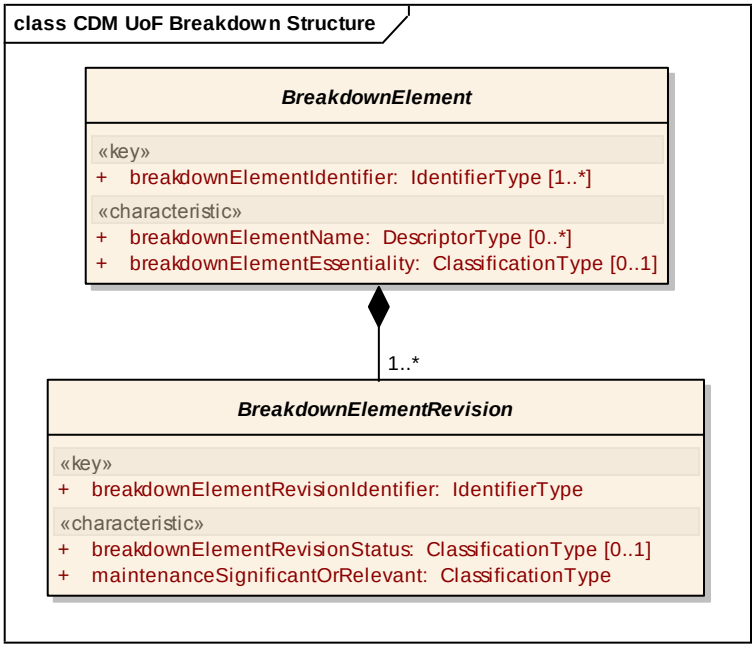
The diagrams contain classes that are both filled and not filled (white). A filled class represents classes that are introduced and defined within the Unit of Functionality (UoF) under consideration. Classes that are not any filled are classes that are defined in other UoFs, and are always presented without any of their attributes.

In, SoftwareElement and SoftwareElementRevision are fully defined in the UoF Software Element. However, BreakdownElement and BreakdownElementRevision are defined in another UoF Breakdown Structure.



ICN-B6865-SX004G0016-001-00

Fig 16 Example of a class defined in another UoF



ICN-B6865-SX004G0017-001-00

Fig 17 Actual definition of the classes

3 S-Series Primitives (Data Types)

3.1 Overall description

The data types used in the ILS S-Series 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 data types (primitives) are used. The primitives used, have been defined by the ASD/AIA DMEWG.

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

However, in the S-Series, this is of the primitive type “IdentifierType”. In this way, the string value representing the part number must always be defined together with the following attributes:

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

The following list the <<primitives>> used throughout the S-Series ILS specification’s data models. For details, refer to the SX001G and SX002D.

Note

UML primitives have been used to define attribute types for the <<primitive>> attributes. There are also some exceptions in the S-Series ILS specifications data models where UML primitives have also been used to define the attribute type for class attributes.

3.2 Organization

This <<primitive>> is used to represent an organization (eg, a company or an agency) using its identifier and a name.

3.3 Date Time Type

This <<primitive>> is used to represent date and time information, using hours/months/days/hours/minutes/seconds. It provides the possibility of defining an offset.

3.4 IdentifierType

This <<primitive>> is used to represent any form of identification, including names. Therefore it uses an identification character set, a classifier of the identifier and an organization that owns/sets the identification.

Note

Identifiers may be used in the class definitions to support one to many identifiers of the eg partIdentifier.

3.5 DescriptorType

This <<primitive>> is used to represent any textual description. It provides information about the organization that provided the description and the date when the description was assigned.

Note

Descriptor elements may be used to support one to many text values of the attribute,

typically used of alternate language, organization or date representations of nomenclatures or descriptions.

3.6 ClassificationType

This <<primitive>> is used for representing references to terms, which can be used for classifications. In legacy applications those often appear as value lists (enumerations).

3.7 PropertyType

This <<primitive>> is used for representing property values. A property value can also include information on when a property value was recorded as well as information on the method by which the value has been determined (eg, estimated, or calculated). It also includes the units of measure.

Note:

The Property element may be numeric or character and may be used to support one to many values of the attribute.

4 Compound attributes

The compound attributes define often recurring patterns of attributes that are used together. The following compound attributes exist.

4.1 SerialNumberRange

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

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.

4.3 AuthorizedLife

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

Note

Multiple values for AuthorizedLife mean ‘whichever comes first’.