STEP AP 242 Electrical Harness XML Tutorial & Recommended Practises

Version: 2.0, Draft 1
Date: 2020-11-16
Status: Updated for AP242 IS Second Edition 2020 & upcoming Amendment 2020/2021

Contacts		
Lothar Klein	Sophie Herail	Daniel Ganser
Steinweg 1	CIMPA S.A.S.	Gulfstream Aerospace Corporation
36093 Künzell / Germany	Centreda 1	BTC
	4, Avenue Didier Daurat	171 Crossroads Parkway
	31700 Blagnac, France	Savannah, GA 31407, U.S.A.
	Subcontractor for AIRBUS	
lothar.klein@lksoft.com	Operations SAS – IZMA	dan.ganser@gulfstream.com
	sophie.herail.external@airbus.com	

Links:
https://www.cax-if.org/ewis/ewis_introduction.php
http://www.lotar-international.org/
http://www.ap242.org/
https://en.wikipedia.org/wiki/ISO_10303
http://www.iso.org

Table of Contents

1. Overview

This document focuses on the Electrical Wire Harness capabilities of STEP AP242 edition 2 amendment (1). It explains the concepts and provide recommendations of how to use them. The concepts are defined in a rather generic way that also covers all kinds of electrical installation as well as connectivity for glass fibres, piping and ventilation installations.

Version 1.0 of this document had been based on the initial CD (Committee Draft) of AP242 ed2, and version 1.1 on the DIS (Draft International Standard) and FDIS (Final Draft International Standard). This version is updated for the upcoming (first) amendment of AP242 ed2 and extended for recommended practises. As such this document is an extension of the "Recommended Practices for AP242 Business Object Model XML Assembly Structure". Readers are expected to be familiar with that underlying document. Only differences and extensions will be highlighted here.

This document explains on how to do data exchange via XML, based on the Domain Model (DO-Model). The Domain Model in AP242 ed2 is an upward compatible extension of the Business Object Model (BO-Model) of AP242 ed1.The Domain Model is primarily defined in SysML, but a corresponding Express (ISO 10303-11) representation is also provided. From the Domain Model an XML schema has been derived that defines the structure of the XML files to exchange. The examples provided in this document are based on this XML Schema.

The standard provides a formal mapping from the Domain Model to the Application Reference Model (ARM), and from there to the Application Integrated Model (AIM) of STEP. The AIM is an alternative way on how to implement data exchange and integration for electrical wire harnesses, using either STEP-Part21 files (ISO 10303-21) or other implementation methods such as SDAI (Standard Data Access Interface (ISO 10303-22).

Formal References:

- Recommended Practices for AP242 Business Object Model XML Assembly Structure
  Release 2.1; 2019-12-20; on: www-cax-if.orgPlease refer to that document for all general concepts.
  Note that release 2.1 is based on the first edition AP242 (technical corrigendum).
- STEP: ISO 10303 "Industrial automation systems and integration -- Product data representation and exchange"
  - AP242 ed2: ISO 10303-242:2019: Application protocol: Managed model-based 3D engineering"
  - XSD of AP242 ed2 Amendment, derived from the Domain Model documented in SysML
    Note: This XSD is the basis for this tutorial.
  - the SMRL, "STEP Module and Resource Library", version 9? that contains the ARM (Application Reference Model) and mapping to the MIM (Modular Integrated Model).
    Note: This tutorial does not address on how to implement EWH on the MIM level using p21
- ISO 13584-42:2010 Industrial automation systems and integration — Parts library — Part 42: Description methodology: Methodology for structuring parts families

The work is performed for the joint ASD-Stan / AIA / ProSTEP iViP / PDES Inc. consortium "LOTAR International".

2. Introduction & Use Cases

2.1. New capabilities in AP242 edition 2

With the second edition of STEP AP242 it's capabilities are extended for the representation of electrical wire harness as it is used in aircrafts, vehicles, electronic devices such as computers, machines. In addition the overall connectivity in these vehicles can be represented with all the connected devices. Because of this AP242 ed2 is also suitable for the representation of electrical installation in buildings. Compared to the first edition, the second edition of STEP AP242 has new or extended capabilities for:

- a generic model for shape feature templates and patterns, how they are decomposed, and how a pair of these shape features fit together. This representation is independent of a particular product. With this capability it is possible to represent the typical shape of connectors and contacts and how a pair of them fit together. Note that in STEP a shape and it's aspects can be identified without having a corresponding geometric model in 2D or 3D at hand;
- a generic model for physical connectivity, consisting of contact features and terminals, transport features, assembly joints and constraints and overall connectivity definitions of parts. This model supports the basic Kirchhoff's laws, but also supports a nested hierarchy of assemblies with sub-assemblies. The model is not only suitable for electrical connectivity but is also suitable for thermal, optical, matter (water, air, ...) and other domains;
- representation of electrical devices with all their external contacts. An electrical device can be represented as either a specific part that has a part number and where details of the design are known, or alternatively as a particular configuration of a product class as it can be found in catalogues or standards and for which many different parts from different vendors exist.
- representation of extruded flexible raw products of arbitrary length with a constant cross-section that might be decomposed in a hierarchical way of different materials. This model is suitable to describe wires, cables, fibres, pipes etc. The identification of conductors and shields and various kind of isolators, coatings, coverings etc. is possible. The raw product can be cut into identified pieces of a particular length, called occurrences, and then be used in an assembly like occurrences of other piece parts.
- representation of the topological structure of an electrical wire harness consisting of harness nodes and harness segments. While the harness segments have a dedicated length, but are typically flexible, on the harness nodes rigid materials such as connectors or mounts can be associated. A harness node can be associated to a particular position and orientation in 2D or 3D geometric space, while a harness segment can be associated to a geometric curve. Different 2D and 3D geometric models can be associated to a harness to represent the shape on a foam board, or how the harness is delivered and how it is installed;
- breakdown of a harness into sub-harnesses that can be manufactured independently from each other and that are only assembled into a single harness during the installation process;
- logical splitting of a complete harness into partial harness views for different design teams that are responsible for different regions of the harness.

2.2. Use cases

The wiring harness extension of STEP AP242 edition 2 supports among many others the following use cases:

- exchange of part libraries of wires, cables, connectors, contact and devices;
- exchange of "wire list" of a wiring harness that identifies each wire and to which connector contacts it is joined;
- exchange of "component-list" of a wiring harness assembly, containing all the connectors with associated materials, terminal lugs and blocks, electrical shields and tape marking;

- exchange of the topology of the harness, including the harness segments and how the harness is protected and prepared for installation with grommet, retention clamps, sleeving, overall shielding and others. This can be associated with 2D and 3D geometry for stickline and formboard representation, and final geometry when installed;
- integration of all the aspects of a wiring harness coming from different systems in a database system.
- usage of an electrical wire harness as an occurrence in a higher level assembly such as aircraft or vehicle

2.3. XML File Format, XML Schema

Every AP242 XML file has to call out the XML-Schema of AP242. For the amendment of the 2nd edition of AP242 this is defined as “AP Domain XML schema definition” in the 3rd edition of part ISO/TS 10303-4442:2020.

The “Unit of serialization” (Uos) element is the top most element that encloses all needed schema definitions

All the application objects (AO, entities) not being declared as contained in any other AO are defined as a subtype of cmn:BaseRootObject and show up within the cmn:DataContainer

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<n0:Uos
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:cmn="http://standards.iso.org/iso/ts/10303/-3000/-ed-1/tech/xml-schema/common"
  xmlns:n0="http://standards.iso.org/iso/ts/10303/-4442/-ed-1/tech/xml-schema/
domain_model"
  xsi:schemaLocation="http://standards.iso.org/iso/ts/10303/-4442/-ed-1/tech/xml-
schema/domain_model DomainModel.xsd">
<Header/>
  <DataContainer xsi:type="n0:AP242DataContainer">

  <!--application data-->

  </DataContainer>
</n0:Uos>
```

3. Examples

This tutorial comes together with reference examples that had been worked out by the electrical harness working group of the AP242 Edition 2 development team. The examples are hand-crafted and do not represent any complete design. The intention of the examples is to demonstrate how the different AP242 edition 2 extensions for electrical harness design play together.

The examples are build up onto each other in a few variants to show the capabilities of the standard. Central for the examples are the topological structure of wiring harness H1 and a modular and configurable connector of type ARINC 600 that are widely used in aerospace industry.

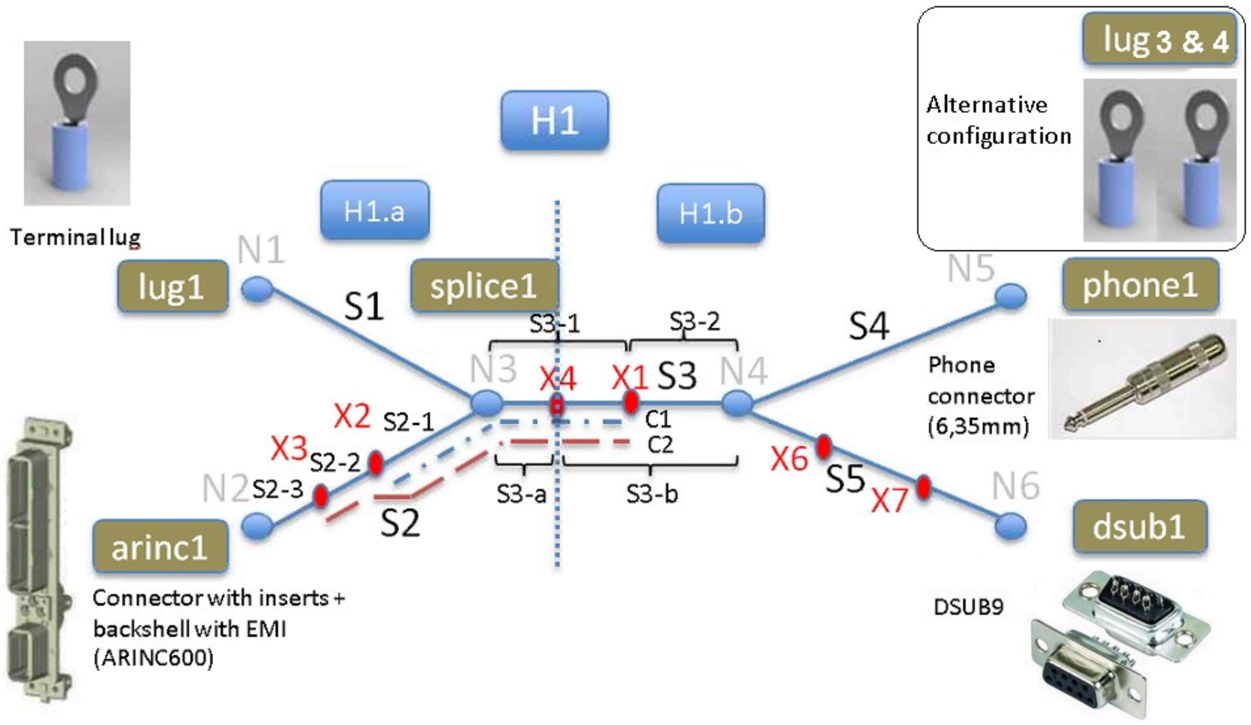


Figure 1: Topological structure and connectors of the provided example Part_H1

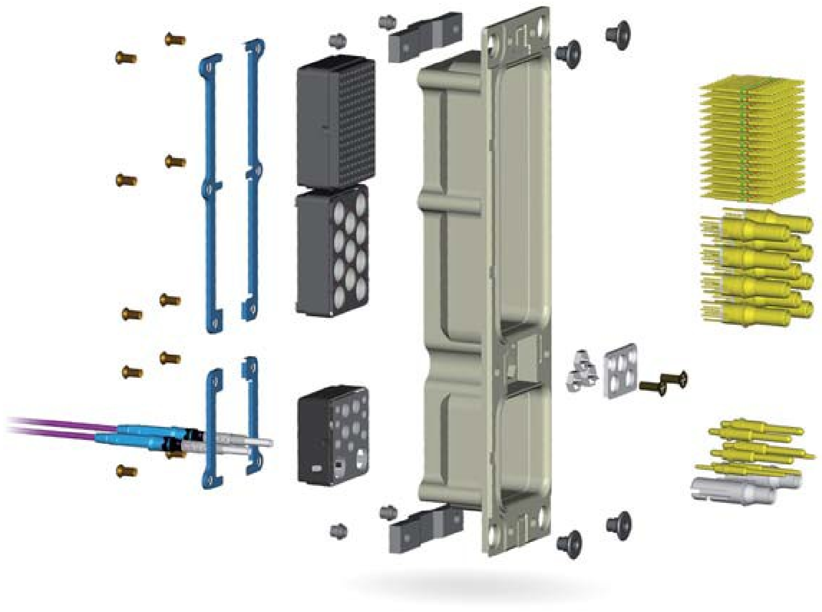


Figure 2: Modular ARINC 600 connector with different kinds of inserts and contacts

Three example XML files are provided that are in detail explained in the following sections.

- HarnessExample_flat.xml
  This file contains the wiring harness H1 with all components are assembled together in a single assembly, including the components the ARINC600 connector is build up
- HarnessExample_hierachical.xml
  This file contains the wiring harness H1 in a hierarchical assembly to represent an artificial assembly structure of the ARINC600 connector. In addition this file contains a second wiring harness H2 that is combined with H1 in an upper level assembly for the whole aircraft. The H1 harness is provided in two different configurations for either a single connector phone1 or two terminal lugs (lug3, lug4).
- HarnessExample_DesignSplitting.xml
  This file contains three wiring harnesses, the sub-harnesses H1a and H1b that are constituents of the overall harness H1.

3.1. Flexible structure of harness H1

The main harness topology in this example consists of five segments S1 to S5 that connects six nodes N1 to N6. On the end-nodes N1, N2, N5 and N6 different kinds of connectors and a terminal lugs are placed.

- Most wires and cables are connected to the ARINC600 connector at node N2 and go through segment S2. These are two coax RG 58 cables (cable1, cable2), a speaker cable (cable3), and a single wire (wire2).
- On the internal node N3 a splice is placed that is realizing a T connection between 3 wires (wire1, wire2, wire3).
- The terminal lug at node N1 is connected by a single wire arriving from segment S1. As S1 contains only a single component no grouping is needed for this segment.
- The phone connector at node N5 is connected by a single speaker wire (cable3) arriving through segment S4. As S4 contains only a single component no grouping is needed for this segment.
- The DSUB connector at node N6 is connected by two coax cables (cable1, cable2) and a wire

(wire3). These three wires and cables are grouped together with lacing in Segment S5.

- In segment S2 wire2 and cable3 are grouped by twisting around each other. The result is then grouped together with cable1 and cable2 and covered by a shielding braid that is connected to the GND of the backshell of the ARINC600 connector.
- Segment S3 contains the same arrangement of cables and wires as in segment S2, but because of the splice at node N3, it contains wire3 instead of wire1.
- A path through the segments S2, S3 and S4 has been defined for the speaker wire (cable 3)
- A path through the segments S2, S3 and S5 has been defined for the two coax cables (cable1, cable2) and the alternative of wire1 respectively wire3

Occurrence	- name	Part-name	Segment / Path	Start-node	End-node
_201004	wire1	WIRE,ELEC,COMP, SNGL CONDUCTOR,150 DEG C	S1	N1	N3
_201104	wire2		S2	N2	N3
_201204	wire3		S3-S5	N3	N6
_202006	cable1	RG 58	S2-S3-S5	N2	N6
_202106	cable2	RG 58	S2-S3-S5	N2	N6
_204006	cable3	Speaker wire	S2-S3-S4	N2	N5
_220005	braid1	Braid 1/2inch	S2-S3	N2	N4
_221005	wrap1	1/2 inch Spiral Wrap - 100 foot spool - Black	C1 = S2-1 + S3-1	X2	X1
_222005	heatshrink1	Shrinkflex Polyolefin Heatshrink Tubing - 4/1 - 25mm 1	C2 = S2-2 + S2-2 + S3-1	X3	X1

Table 1: Wire / cable list for H1

A more detailed sub-topology on the main topology is available through the sub-nodes X1 to X7 that are placed on the main segments S2, S3 and S5. These sub-nodes divide the main segments. Segment S2 is divided by X2 and X3 into S2-1, S2-2 and S2-3. Segment S3 is divided in two alternative ways, S3-1 and S3-2 by X1 and S3-a and S3-B by X4. No sub-segments have been defined for the sub-nodes X6 and X7 on S5 because there had been no need for it. These detailed topology is used as follows:

- to define a path consisting of the segments S2-1 and S3-1 for a protective covering (wrap1)
- to define a path consisting of the segments S2-2, S2-1 and S3-1 for a heatshrink (heatshrink1)
- to split the overall harness H1 into two sub-harnesses H1.a and H1.b that different design groups are responsible for
- to define fixing locations at nodes X6 and X7

3.2. Rigid parts for harness H1

Note: for the purpose of this document we use the term part for those things that we can buy or build several times, while the term component is used for an occurrence of a part within an assembly. An assembly may contain two components (more precisely occurrences) of a part “X” that might be identified as X1 and X2.

Data exchange for electrical wire harness is often restricted to those rigid components that are electrical relevant. All the other “auxiliary” mechanical parts like screws are not part of the data exchange. However for a complete representation of the design they are of course needed. It is up to the particular use case which components are exchanged, and this can be indicated by the specified ApplicationDomains (see clause 5.1).

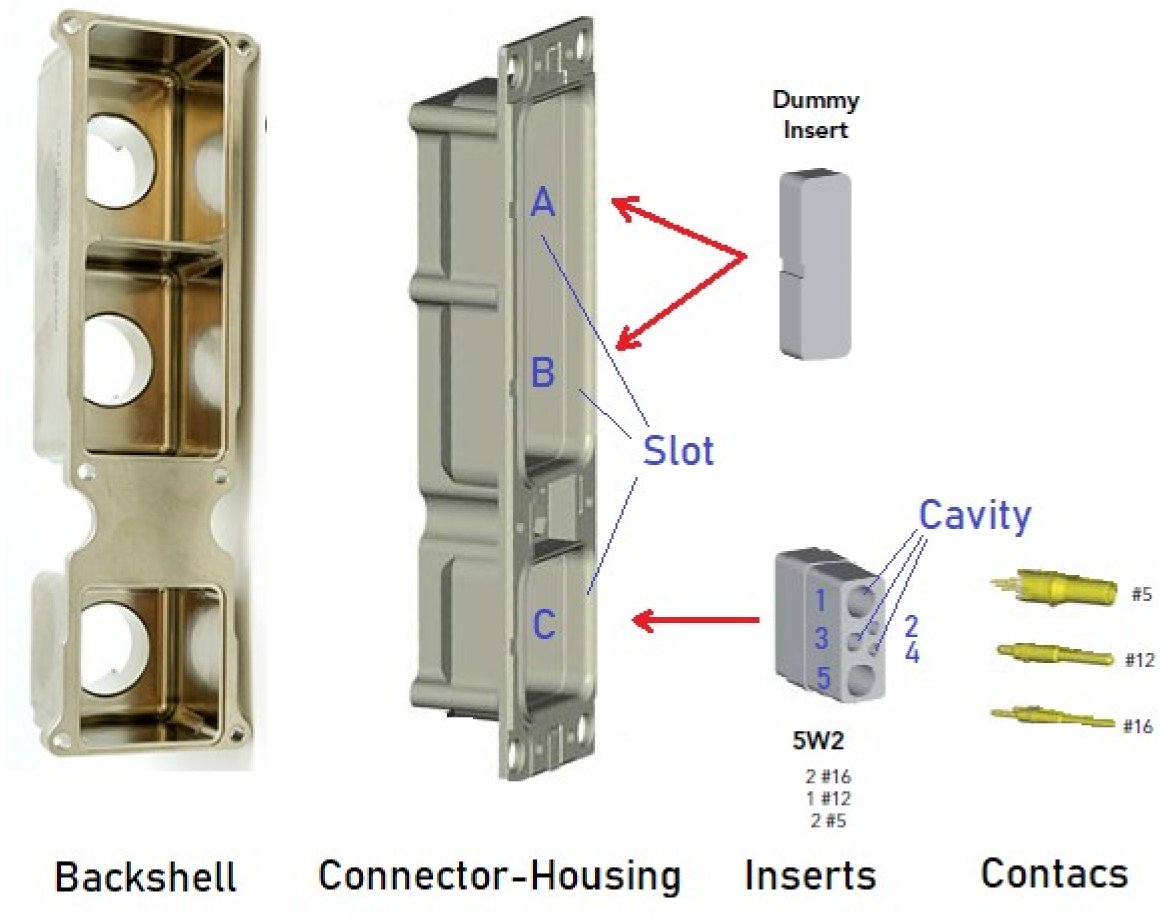


Figure 3: Composition of ARINC600 connector for H1

For the Harness H1, the ARINC600 connector is composed of connector housing of shell size 1, an electrified backshell, two dummy inserts for the A and B slot, and a 5W2 insert for the C slot. The 5W2

Part	Version	View	Id	Name	Category/type	Remark
_101000	_101001	_101002	04034-22-9	WIRE,ELEC,COMP,SNGL CONDUCTOR,150 DEG C	wire, raw material by length	
_102000	_102001	_102002	RG 58	RG 58	cable, raw material by length	
_103000	_103001	_103002	MS5036-153	TERMINAL LUG CRIMP STYLE COPPER INSULATED RING TONGUE	terminal lug	
_104000	_104001	_104002	16 AWG	Speaker wire	cable, raw material by length	
_110000	_110001	_110002	SB6 4 1 M G 05 W2 P E1 01 AA	ARINC 600 set	connector kit	not in flat example
_111000	_111001	_111002	8660-31A-100-01A/AA	ARINC 600 shell 1 A and B dummy	connector insert	
_112000	_112001	_112002	8660-5W2	ARINC 600 shell 1 C 5W2 insert	connector insert	
_113000	_113001	_113002	8660-2485	#5 Coax contact	connector contact	
_114000	_114002	_103090	8660-249	#16 Rack plug power contact	connector contact	
_115000	_115001	_115002	8660-5W2x1	Insert 5W2 assembly	connector insert	not in flat example
_116000	_116001	_116002	8660-250	#12 Rack plug signal contact	connector contact	
_117000	_117001	_117002	TM2PB	Phone connector 6.35mm	connector	
_118000	_118001	_118002	M81824	Butt Splice Terminal	splice	
_119000	_119001	_119002	DEP09S065TLF	D-sub 9 Pin Db9 Female Solder Type Socket Connector	connector	
_120000	_120001	_120002	TC122	Braid 1/2inch	overbraid	
_121000	_121001	_121002	F6W1.50BK	Wrap	protective covering	
_122000	_122001	_122002	H4N1.00BK	HeatShrink	protective covering	
_123000	_123001	_123002	8660-140	Backshell EMI connector	electrified backshell	
_124000	_124001	_124002	Battery-Std	Battery,12V,100Ah	piece part	
_125000	_125001	_125002	SB6 4 1 M	ARINC 600 Shell Size 1 Rack Plug	connector_housing	flat example only
_311000	_311001	_311002	Part_H1	Electrical Harness team reference example	wiring harness	main H1 part
_411000	_411001	_411002	Part_H2	Electrical Harness example 2 (minmal)	wiring harness	not in H1
_511000	_511001	_511002	Aircraft99x	Aircraft99x		not in H1

Table 2: Parts used in the provided examples

insert will hold simple contacts of #5, #12 and #16 size in the cavities 1 to 5.

The elements caa data exchange for electrical harness is often limited to

The electrical relevant components

3.3. Connectivity of harness H1

3.4. Flat assembly structure for harness H1

3.5. Hierarchical assembly structure for Harness H1

3.6. Harness H1 with assembly alternatives

Same as Example 3.1 and Figure 2, but now with alternative of lug3 & 4 instead of phone 1.

One way to exchange overlapping alternative assemblies in STEP is to use a so called 150% structure. So the design of H1 contains all components, including phone1, lug3 and lug4. However a single 100% structure can only contain either phone1 or otherwise lug3 and lug4. This is realised by the two ProductConfiguration "H1 stereo plug" and H1 hookup lugs" for the same ProductConcept. With ConfiguredAssemblyEffectivity the not used Occurrencess and PartShapeElements are excluded.

No further details of this specific use case is explained here, but the example structures can be found in HarnessExample.xml file.

3.7. Example Harness H1 designed by two teams as H1a and H1b

Same as Example 3.1 and Figure 1, but now split into an H1.a and a H1.b view, developed by different teams.

For the exchange of such a scenario three different instances of WiringHarnessAssemblyDesign are created, one for H1 with the PredefinedApplicationDomainEnum value "complete_design" and two others for H1.a and H1.b with the PredefinedApplicationDomainEnum value "partial_design". The partial design views are related with the complete design by instances of DefinitionalPartViewUsage. By this the complete design H1 is automatically defined being the merger of H1a and H1b.

In the similar way there are three EdgeBasedTopologicalRepresentationWithLengthConstraint with the same RepresentationContext, one for the complete Harness H1 and two others for the partial harnesses H1a and H1b. They all share the same instance of Vertex X4, and so are explicitly linked together.

No further details of this specific use case is explained here, but the example structures can be found in HarnessExample_DesignSplitting.xml file.

3.8. Higher Level Assembly with harness components H1 and H2

Example: Aircraft with two harnesses

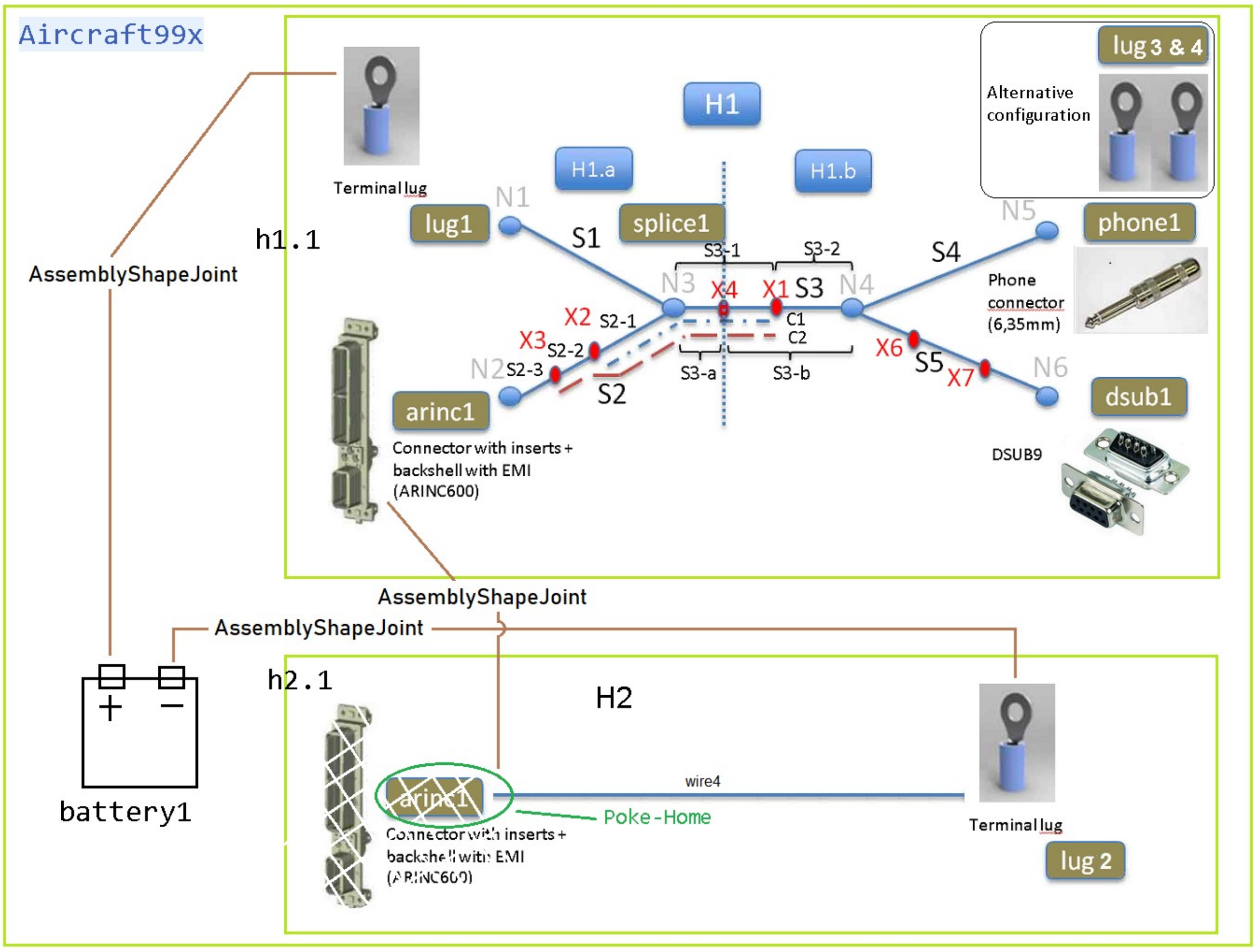


Figure 4: Harnesses H1 and H2 combined in top assembly Aircraft99x

The example also contains a second mini harness example Part_H2 and a top assembly Aircraft99x (figure 4). The mini harness example Part_H2 consists only of a single wire (wire4) and a terminal lug (lug2) to which one end of the wire is connected to. The other end of the wire is intended to be connected to the arinc1 connector from the Part_H1 example above, but this connector is not part of the H2 assembly. This situation is in some industries called “poke-home”.

The top assembly Aircraft99x that contains 3 Occurrences:

- h1.1 that is of type Part_H1
- h2.1 that is of type Part_H2
- battery1 that is connected with the interface terminals of the two terminal lugs

The PLUS and MINUS terminals of the battery is connected to the interface terminals of the two terminal lugs (lug1 and lug2).

Also the “poke-home” connection from the H2 assembly can finally be connected in that top assembly because arinc1 from Part_H1 and wire4 from Part_H2 are available in this assembly.

In the following clauses this example is further detailed.

Occurrence	Name	Occurrence-Terminal	Name	Assembly Join
	Part_H1			
_201004	wire1	_201006	end a	_311010-1
		_201007	end b	_311050-1
_201104	wire2	_201106	end a	_311040-1
		_201107	end b	_311050-2
_201204	wire3	_201206	end a	_311052-2
		_201207	end b	_311034-2
_204006	cable3	_204013	end a A	_311045-1
		_204014	end a B	_311046-1
		_204023	end b A	_311020-1
		_204024	end b B	_311021-1
_202006	cable1	_202013	end a signal	_311041-1
		_202014	end a shield	_311042-1
		_202023	end b signal	_311030-2
		_202024	end b shield	_311031-2
_202106	cable2	_202113	end a signal	_311043-1
		_202114	end a shield	_311044-1
		_202123	end b signal	_311032-2
		_202124	end b shield	_311033-2
_203005	lug1	_203006	External	no
		_203008	Internal	_311010-2
_217100	phone1	_217101	Interface signal	no
		_217102	Join signal	_311020-2
		_217103	Interface gnd	no
		_217104	Join gnd	_311021-2
_218100	splice1	_218103	end a	_311051-2
		_218104	end b	_311052-1
_219100	dsub1	_219113	Interface 1	no
		_219114	Join 1	_311030-1
		_219123	Interface 2	no
		_219124	Join 2	_311031-1
		_219133	Interface 3	no
		_219134	Join 3	_311032-1
		_219143	Interface 4	no
		_219144	Join 4	_311033-1
		_219153	Interface 5	no
		_219154	Join 5	_311034-1
		_219163	Interface 6	no
		_219164	Join 6	n.c.
		_219173	Interface 7	no
		_219174	Join 7	n.c.
		_219183	Interface 8	no
		_219184	Join 8	n.c.
		_219193	Interface 9	no
		_219194	Join 9	n.c.

Occurrence	**Name**	**Occurrence-Terminal**	**Name**	**Assembly Join**
Part_H1 (continued)				
_213100	arinc1/C-Assy/coax1	_213121	Interface signal	no
		_213122	Join signal	_311041-2
		_213123	Interface gnd	no
		_213124	Join gnd	_311042-2
_213210	arinc1/C-Assy/coax2	_213221	Interface signal	no
		_213222	Join signal	_311043-2
		_213223	Interface gnd	no
		_213224	Join gnd	_311044-2
_214110	arinc1/C-Assy/power3	_214201	Int term	no
		_214223	Join term	_311045-2
_214220	arinc1/C-Assy/power4	_214201	Int term	no
		_214224	Join term	_311046-2
_216120	arinc1/C-Assy/signal5	_216101	Int term	no
		_216122	Join term	_311040-2
_223200	arinc1/backshell1	_223201	GND1	_311047-1
_220005	braid1	_220007	shield-A	_311047-2
Part_H2				
_203005	lug2	_203006	External	no
		_203008	Internal	_411010-1
_201204	wire4	_201206	end a	(_411011-1)
		_201207	end b	_411010-2

Table 3: AssemblyShapeJoints

4. Part, PartVersion, Part category

Objects of type **Part** are used for the all the parts that are used to build an electrical harness. In addition an electrical harness is modelled as *Part* as well, so that it can be used in a next higher assembly; e.g. for a complete aircraft. For each Part at least one PartVersion shall be defined. A PartVersion may have one or several views (see next chapter).

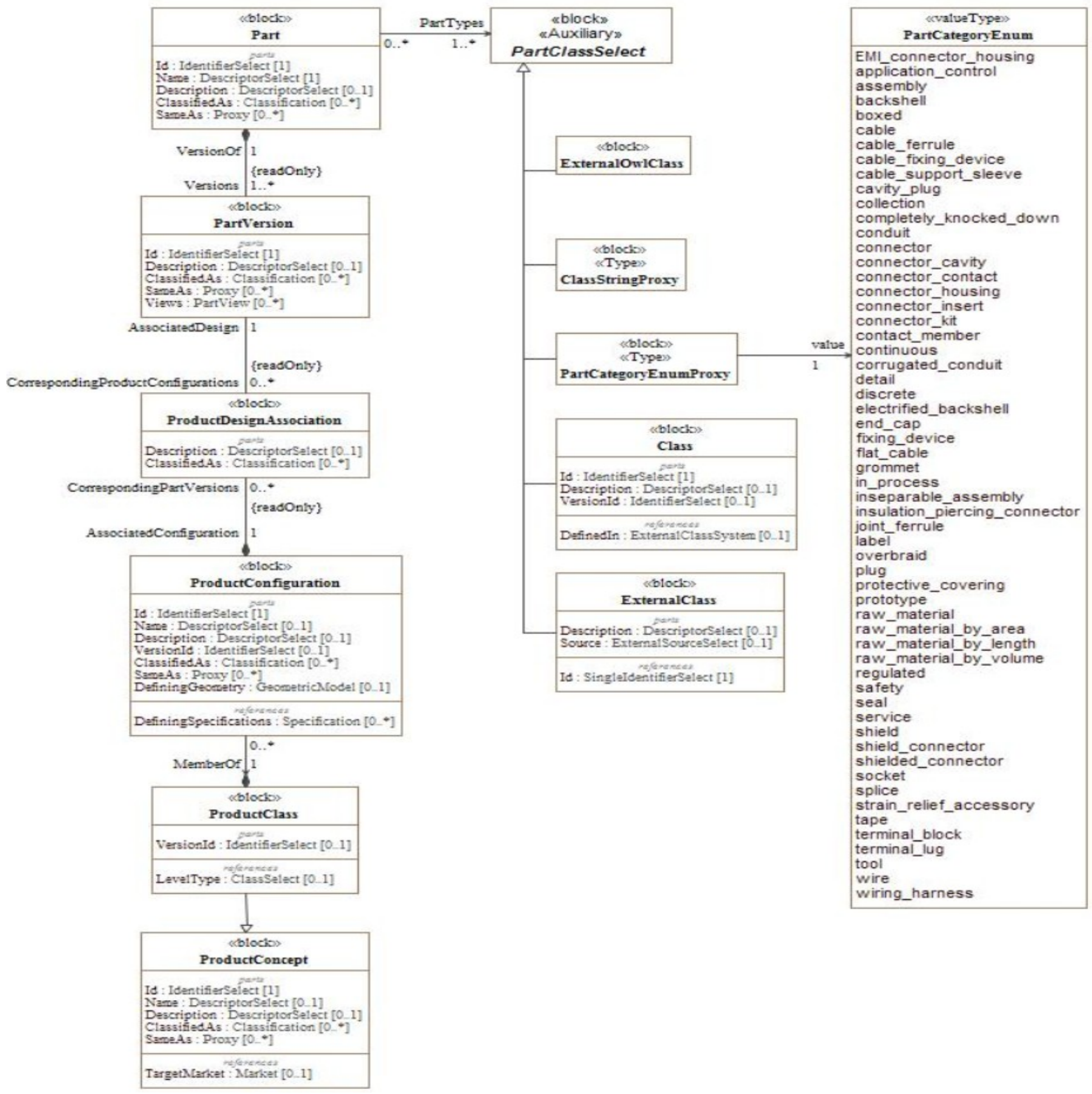

«block» Part
parts
Id : IdentifierSelect [1]
Name : DescriptorSelect [1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
SameAs : Proxy [0..*]
PartTypes
0..*
1..*
«block» «Auxiliary» PartClassSelect
VersionOf 1
{readOnly}
Versions 1..*
«block» PartVersion
parts
Id : IdentifierSelect [1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
SameAs : Proxy [0..*]
Views : PartView [0..*]
AssociatedDesign 1
{readOnly}
CorrespondingProductConfigurations 0..*
«block» ProductDesignAssociation
parts
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
CorrespondingPartVersions 0..*
{readOnly}
AssociatedConfiguration 1
«block» ProductConfiguration
parts
Id : IdentifierSelect [1]
Name : DescriptorSelect [0..1]
Description : DescriptorSelect [0..1]
VersionId : IdentifierSelect [0..1]
ClassifiedAs : Classification [0..*]
SameAs : Proxy [0..*]
DefiningGeometry : GeometricModel [0..1]
references
DefiningSpecifications : Specification [0..*]
0..*
MemberOf 1
«block» ProductClass
parts
VersionId : IdentifierSelect [0..1]
references
LevelType : ClassSelect [0..1]
«block» ProductConcept
parts
Id : IdentifierSelect [1]
Name : DescriptorSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
SameAs : Proxy [0..*]
references
TargetMarket : Market [0..1]
«block» ExternalOwlClass
«block» «Type» ClassStringProxy
«block» «Type» PartCategoryEnumProxy
value
1
«block» Class
parts
Id : IdentifierSelect [1]
Description : DescriptorSelect [0..1]
VersionId : IdentifierSelect [0..1]
references
DefinedIn : ExternalClassSystem [0..1]
«block» ExternalClass
parts
Description : DescriptorSelect [0..1]
Source : ExternalSourceSelect [0..1]
references
Id : SingleIdentifierSelect [1]
«valueType» PartCategoryEnum
EMI_connector_housing
application_control
assembly
backshell
boxed
cable
cable_ferrule
cable_fixing_device
cable_support_sleeve
cavity_plug
collection
completely_knocked_down
conduit
connector
connector_cavity
connector_contact
connector_housing
connector_insert
connector_kit
contact_member
continuous
corrugated_conduit
detail
discrete
electrified_backshell
end_cap
fixing_device
flat_cable
grommet
in_process
inseparable_assembly
insulation_piercing_connector
joint_ferrule
label
overbraid
plug
protective_covering
prototype
raw_material
raw_material_by_area
raw_material_by_length
raw_material_by_volume
regulated
safety
seal
service
shield
shield_connector
shielded_connector
socket
splice
strain_relief_accessory
tape
terminal_block
terminal_lug
tool
wire
wiring_harness

Figure 5: Part, PartVersion and categories for Parts

Example: A wire part that is provided as bulk material. For the usage of the *Part* in an *Occurrence* it needs to be cut to a specific length.

```
<Part uid="_101000">
  <Id id="04034-22-9"/>
  <Name>
```

```
    <CharacterString>WIRE,ELEC,COMP,SNGL CONDUCTOR,150 DEG C</CharacterString>
  </Name>
  <PartTypes>
    <PartCategoryEnum>wire</PartCategoryEnum>
    <PartCategoryEnum>raw_material_by_length</PartCategoryEnum>
  </PartTypes>
  <Versions>
    <PartVersion uid="_101001">
      <Id id="Version 1"/>
        <Views>
        ....
        </Views>
    </PartVersion>
  </Versions>
</Part>
```

The example provided by this tutorial contains the following Parts:

```
<Part uid="_101000"> <!-- WIRE,ELEC,COMP,SNGL CONDUCTOR,150 DEG C -->
<Part uid="_102000"> <!-- RG 58 (COAX cable) -->
<Part uid="_103000"> <!-- TERMINAL LUG CRIMP STYLE COPPER INSULATED RING TONGUE -->
<Part uid="_104000"> <!-- Speaker wire -->
<Part uid="_117000"> <!-- Phone connector 6.35mm -->
<Part uid="_118000"> <!-- Splice-->
<Part uid="_119000"> <!-- Dsub9-->
<Part uid="_110000"> <!-- ARINC 600-->
<Part uid="_111000"> <!-- ARINC 600 shell 1 A and B dummy -->
<Part uid="_112000"> <!-- ARINC 600 shell 1 C 5W2 insert -->
<Part uid="_113000"> <!-- #5 Coax contact -->
<Part uid="_114000"> <!-- #16 Rack plug power contact -->
<Part uid="_115000"> <!-- Insert 5W2 assembly -->
<Part uid="_116000"> <!-- 12 Rack plug signal contact -->
<Part uid="_123000"> <!-- Backshell EMI connector -->
<Part uid="_120000"> <!-- Braid 1/2inch -->
<Part uid="_121000"> <!-- Wrap -->
<Part uid="_122000"> <!-- HeatShrink -->
<Part uid="_311000"> <!-- Part H1 -->
```

The main attributes for a Part are its Id (used as master part number without version information) and Name. For the design exchange of electrical harness the specification of a part category is essential so that a receiving system can filter the data in a certain way. In this example the pre-defined category “wire” is used.

4.1. Generic Predefined Part Categories

AP242 defines the following generic categories (not specific for electrical harness):

- **application control**: this type of classification is used to indicate that a part shall be considered under certification aspects
  EXAMPLE Prior to the release of a new car to the market, both function and quality of certain parts have to be certified by an authority, e.g., the department of transportation. For such item objects, certification requirements have to be considered during the design phase.
- **assembly**: this type of classification shall be used for any part that has an *AssemblyDefinition* provided for at least one of its versions, i.e., it is decomposed further;
- **boxed**: specifies the role of the boxed part;
  EXAMPLE A can with one litre of a specific oil.
- **collection**: this type of classification shall be used for any part that has a *CollectionDefinition* provided for at least one of its versions;

- **completely knocked down**: this type of classification is used to indicate that a part is used in a production site that has assembling facilities only; EXAMPLE The 'completely knocked down' may indicate that the components are shipped to and assembled in a different country.
- **continuous**: this type of classification is used to indicate the part that can be measured by its volume or weight;
- **detail**: this type of classification shall be used for any part that has no *AssemblyDefinition* provided for any of its versions, i.e., it is not further decomposed;
- **discrete**: this type of classification is used to indicate that a part is one single piece.
- **in process**: this type of classification is used to indicate that the part identifies an intermediate object in a manufacturing process; NOTE Detailed information about the stage the part has within the manufacturing process may be obtained from the *ProcessOperationOccurrence*.
- **inseparable assembly**: the part plays the role of an inseparable assembly;
- **prototype**: this type of classification is used to indicate that the part identifies a prototype and is not intended for serial production;
- **raw material**: the part plays the role of raw material;
- **raw material by area**: the part plays the role of raw material, cut by area, e.g. chipboard;
- **raw material by length**: the part plays the role of raw material, cut by length, e.g. cable;
- **raw material by volume**: the part plays the role of raw material, cut by volume, e.g. a raw block of marble or oil;
- **regulated**: this type of classification is used to indicate that for a part certain regulations have to be considered;
- **safety**: this type of classification is used to indicate that a part is relevant for safety purposes;
- **service**: this type of classification is used to indicate that a part is relevant for service purposes;
- **tool**: the part plays the role of a tool.

Recommendation: For the purpose of an electrical wire harness design, most Parts require a part category of either “**discrete**” or “**raw material by length**”. In special cases “**raw material by area**” or “**raw material by volume**” might be used alternatively.

4.2. Part and ProductClass Categories specific for Electrical Harness and other electrical applications

Here the pre-defined categories for electrical harness and other electrical applications. Most of these categories are standardized by IEC in the Electropedia: http://www.electropedia.org/

Preprocessors are required to provide the following part categories if they apply. This is needed so that a post-processor can be configures to act properly on the on the incoming information and sort the parts suitable for the receiving system.

- **wiring harness**: assembly with a harness topology, consisting of cables or wires to enable electrical or optical connectivity, grouped together in one or more harness segments, each between two harness nodes in which the cables or wires either switch to other harness nodes or in which the cables and wires ends in connectors, contact members or terminal lugs
- **wire** [IEV ref 151-12-28]: flexible cylindrical conductor, with or without an insulating covering, the length of which is large with respect to its cross-sectional dimensions
  Note – The cross-section of a wire may have any shape, but the term "wire" is not generally used for ribbons or tapes.

- **cable** [IEV ref 151-12-38]: assembly of one or more conductors and/or optical fibres, with a protective covering and possibly filling, insulating and protective material
- **flat cable** [IEV ref 461-06-05]: multi-core cable having cores or groups of cores arranged in parallel flat formation
- **connector kit**: collection of connector parts that are intended to be assembled together and that contain at least one connector housing and that may contain alternative parts that may or may not be used in the final assembly
- **contact member** [IEV ref 151-12-16]: conductive element intended to make an electric contact
- **connector contact**: contact member that is intended to be contained in a connector
- **connector** [IEV ref 151-12-19]: device providing connection and disconnection to a suitable mating component
  Note – A connector has one or more contact members.
- **plug** [IEV ref 151-12-21]: connector attached to a cable
- **socket** [IEV ref 151-12-20]: connector attached to an apparatus, or to a constructional element or alike
  Note – Contact members of a socket may be socket contacts, pin contacts or both.
- **shielded connector** [IEV ref 581-26-19]: connector designed to prevent the radiation of electromagnetic interference to and from the internal conductor(s)
- **cavity plug**: plug for a connector cavity for the purpose of sealing
- **seal**: mechanical object that helps join other mechanical objects together by preventing leakage, containing pressure, or excluding contamination
- **connector housing** [IEV ref 581-27-10]: part of a connector into which the connector insert and contacts are assembled
- **emi connector housing**: connector housing that shields against electromagnetic interference
- **connector insert** [IEV ref 581-27-11]: insulating element designed to support and position contacts in a connector housing
- **backshell**: connector accessory that is closing a connector from the back side and guide the wires and cables
- **electrified backshell**: backshell that is intended to be conductive
- **cable support sleeve** [IEV ref 581-27-23]: flexible accessory or a part of a component placed around the cable to minimize flexing of the cable at the point of entry into the component
- **grommet** [IEV ref 581-27-19]: part of a component or an accessory, used to support and protect the wires or cable at the point of entry; it may also prevent the ingress of moisture or contaminants
- **strain relief accessory**: connector accessory to guide and provide strain relief to wires and cables
- **protective covering**, sheat (North America jacket) [IEV ref 461-05-03]: uniform and continuous tubular covering of metallic or non-metallic material, generally extruded
  Note – The term sheath is only used for metallic coverings in North America, whereas the term jacket is used for non-metallic coverings.
- **overbraid**: protective covering (sheath) that is also a braid
- **conduit** [IEV ref 442-02-03]: a part of a closed wiring system of generally circular cross section for insulated conductors and/or cables in electrical or communication installations, allowing them to be drawn in and/or replaced
- **corrugated conduit** [IEV ref 442-02-06]: a conduit in which the profile is corrugated in the longitudinal section
  Note – Both annular and helical corrugated conduits are permissible and a combination of both

corrugated and plain conduit is possible.

- **shield** (of a cable) [IEV ref 461-03-04]: surrounding earthed metallic layer which serves to confine the electric field within the cable and/or to protect the cable from external electrical influence
  Note 1 – Metallic sheaths, foils, braids, armours and earthed concentric conductors may also serve as shields.
  Note 2 – In French, the term "blindage" may be used when the main purpose of the screen is the protection from external electrical influence.
- **terminal lug** [IEV ref 461-17-01]: metallic device to connect a cable conductor to other electrical equipment
- **terminal block** [IEV ref 581-26-26]: part of a component or an accessory, used to support and protect the wires or cable at the point of entry; it may also prevent the ingress of moisture or contaminants
- **joint ferrule**, through connector (of cables) [IEV ref 461-17-04]: metallic device for connecting two consecutive lengths of conductor
- **cable ferrule** [IEV ref 581-27-18]: accessory in the form of a short tube to provide cable support or termination for a cable screen
- **shield connector**, screen connector [IEV ref 461-17-12]: device used to make a connection to the screen or shield of a cable for the purpose of continuity or earthing
- **insulation piercing connector** [IEV ref 461-11-08]: connector in which electrical contact with the conductor is made by metallic protrusions which pierce the insulation of the cable core
- **splice** [IEV ref 581-24-19]: connecting device with barrel(s) accommodating conductor(s) with or without additional provision to accommodate and secure the insulation
- **fixing device** [IEV ref 442-02-40]: system component specifically designed to secure other components to the wall, ceiling, floor or other structure
- **cable fixing device**: fixing device for a cable on a structure
- **label**: part that is intended to attach written information to other parts
- **tape** [IEV ref 212-15-03]: sheeting or plastic film of limited width and in long continuous lengths
  Note – The width is typically less than some hundred millimetres.
- **end cap** [IEV ref 461-20-07]: device placed on the ends of a cable to prevent the ingress of moisture during storage, transportation and installation
- **connector cavity**: cavity in a connector, connector housing or insert intended to receive a connector contact or multi contact

4.3. Example part with PartCategories

Here an example of a *Part* with a single *PartVersion* (at least one is mandatory). The part categories tells an importing tool on how to handle the part. In this case we know that it is a **discrete** part, and so it comes in pieces, not in some quantity such as length, area or volume that one would need to specify before it can be used (see subtypes of Occurrence). Another important information is that it is a **connector_contact**; so it is something that is used to assembly a connector and that it is a part that most likely has electrical or optical connectivity functionality.

```xml
<Part uid="_113000">
  <Id id="8660-2485"/>
  <Name> <CharacterString>#5 Coax contact</CharacterString> </Name>
  <PartTypes>
    <PartCategoryEnum>connector_contact</PartCategoryEnum>
    <PartCategoryEnum>discrete</PartCategoryEnum>
  </PartTypes>
  <Versions>
```

```
    <PartVersion uid="_113001">
      <Id id="Version 1"/>
      <Views>
      ...
      </Views>
    </PartVersion>
  </Versions>
</Part>
```

5. PartView and ViewContext

The meaning and structure for *Part*, *PartVersion*, *PartView* with reference to ViewContext and also how to associate a geometric model is explained in great detail in the AP242 BO-Model XML Assembly Structure document and so we do not need to repeat this here.

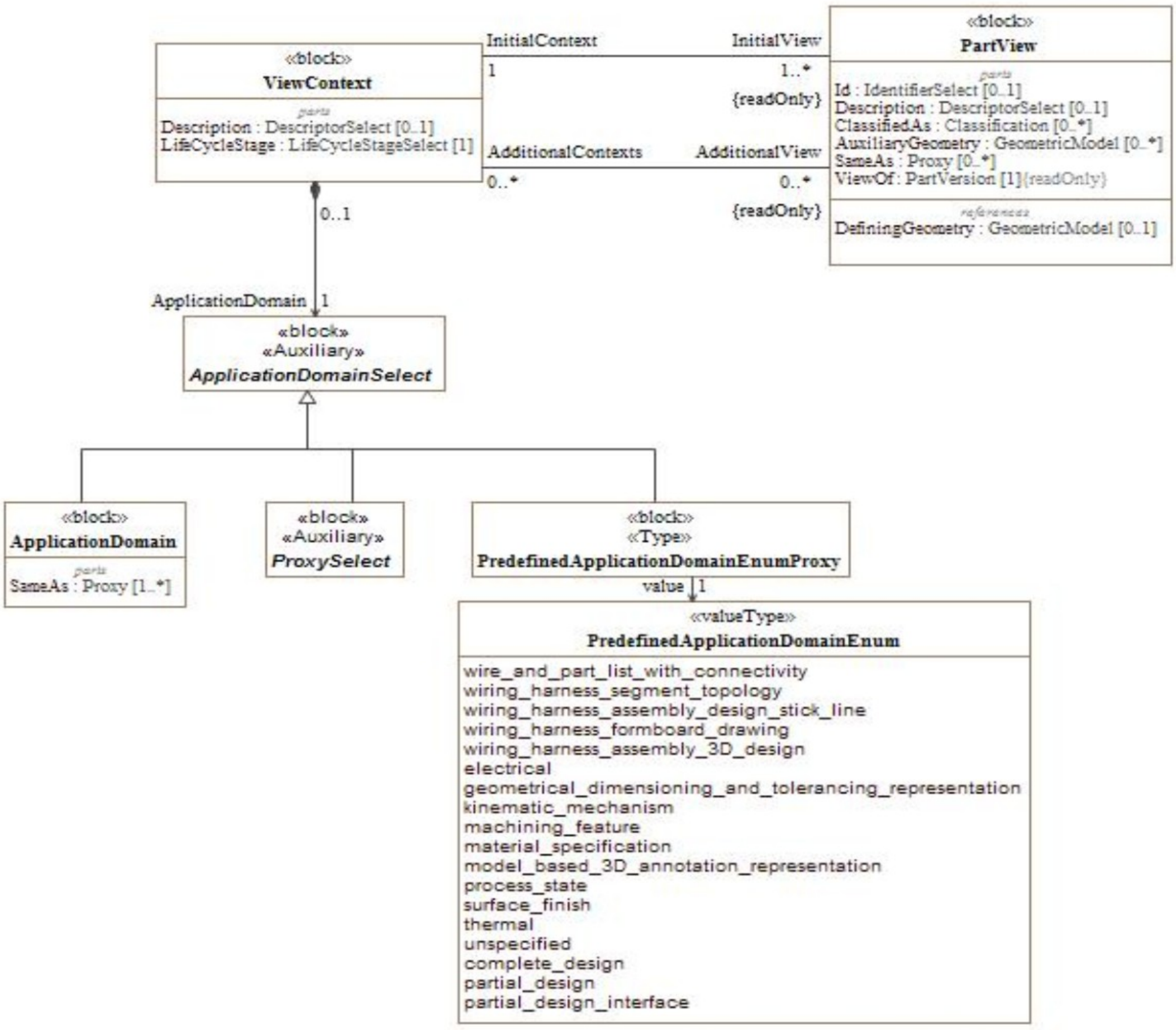

«block»
ViewContext
parts
Description : DescriptorSelect [0..1]
LifeCycleStage : LifeCycleStageSelect [1]
InitialContext
1
InitialView
1..*
{readOnly}
AdditionalContexts
0..*
AdditionalView
0..*
{readOnly}
«block»
PartView
parts
Id : IdentifierSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
AuxiliaryGeometry : GeometricModel [0..*]
SameAs : Proxy [0..*]
ViewOf : PartVersion [1]{readOnly}
references
DefiningGeometry : GeometricModel [0..1]
0..1
ApplicationDomain
1
«block»
«Auxiliary»
ApplicationDomainSelect
«block»
ApplicationDomain
parts
SameAs : Proxy [1..*]
«block»
«Auxiliary»
ProxySelect
«block»
«Type»
PredefinedApplicationDomainEnumProxy
value
1
«valueType»
PredefinedApplicationDomainEnum
wire_and_part_list_with_connectivity
wiring_harness_segment_topology
wiring_harness_assembly_design_stick_line
wiring_harness_formboard_drawing
wiring_harness_assembly_3D_design
electrical
geometrical_dimensioning_and_tolerancing_representation
kinematic_mechanism
machining_feature
material_specification
model_based_3D_annotation_representation
process_state
surface_finish
thermal
unspecified
complete_design
partial_design
partial_design_interface

Figure 6: Data model for PartView and ViewContext

The specifics for electrical harness, compared to the Assembly Structure Recommended Practises, is explained in the following sub-clauses:

5.1. *ViewContext: ApplicationDomain*

A *PartView* shall have at least one *ViewContext* for the *InitialContext* and may have further *ViewContexts* as *AdditionalContexts*. In the recommended practises for assembly design an example of a *ViewContext* with the *ApplicationDomain* "**mechanical design**" and the *LifeCycleStage* "**design**" is provided. This clause defines the values for *ApplicationDomain* and *LifeCycleStage* to be used for the purpose of electrical wire harness.

For the design of an electrical harness the *ViewContexts* of the harness view itself and the ones of all assembled underlying *Part/PartVersion/PartViews* shall have the value "design" for the *LifeCycleStage* attribute.

For the purpose of the design of an electrical harness the *ApplicationDomain* **"electrical"** shall be used for all *PartViews* that define elements relevant for the electrical or optical connectivity such as electrical terminals and nodes. When this *ApplicationDomain* is used for a *PartView* it tells an importing tool that all relevant electrical connectivity information is provided. The *ApplicationDomain* **"electrical"** may be used as either *InitialContext* or *AdditionalContexts*.

As detailed in the assembly recommended practises, an *AssemblyDefinition* is a specialization of a *PartView* that is to be used to define an assembly of parts. In the case that the assembly of an electrical harness is defined, the sub-subtype ***WiringHarnessAssemblyDesign*** shall be used . The following values for the *ApplicationDomain* of additional contexts for a *WiringHarnessAssemblyDesign* are defined:

- electrical: the application domain is electrical. This value should be used for all parts that have electrical relevant content such as terminals (e.g. a connector) or conductors (e.g. a wire).
- the exchange content of a wiring harness assembly view can be stated by these pre-defined values:
    - wiring_harness_assembly_3d_design: a 3D geometry model of the harness topology of a wiring harness assembly.
    - wiring_harness_assembly_design_stick_line: a stick line drawing of a wiring harness assembly.
    - wiring_harness_formboard_drawing: a formboard drawing of a wiring harness assembly.
    - wiring_harness_formboard_drawing: the application domain comprises a formboard drawing of a wiring harness assembly.
    - wiring_harness_segment_topology: the complete harness topology of a wiring harness assembly.

For a simple harness design there is only one PartView of type ***WiringHarnessAssemblyDesign***. However for complex design there might be the need to split a harness into several partial harnesses that are manufactured independently and only combined into a single one during installation. For a rather big harness it might be needed to split the complete harness design into several partial harness designs for which separate teams are responsible; and that may be versioned independently. The model also supports the design process from high level concepts down to detailed designs and be able to trace through the accumulated data from version to version. The completion state of a WiringHarnessAssemblyDesign is indicated by its application domain to cover one or more of these

- complete_design: the PartView that represents a complete design.
- partial_design_interface: the PartView that represents the interface between two or more partial designs. wire_and_part_list_with_connectivity: all wires and wire caps, cables, connectors, and associated material, contact members, terminal lugs, terminal blocks, splices and electrically non-conductive through termination, electrical shielding definition in the assembly, tape marking of a wiring harness assembly.
- partial_design_interface: the PartView that represents the interface between two or more partial designs.

Harness Structures:
xxxx Provide only references to later clause on HarnesSegment, HarnesNode, Connectivity, topology ....

An electrical wire harness is primarily build up of flexible segments that contain the wire and cables and that are connected by branch nodes and extremity nodes for the connectors. The data structure in AP242 is a bit more complex than this as the use cases also require to represent partial harnesses and the way harnesses are built up and installed later on. To support this (and as a general STEP approach) the HarnessSegments and HarnessNodes are specific to a single WiringHarnessAssemblyDesign and reference to a topological harness model that can be shared by several partial designs or designs representing different states of the development process. The topological model consists primarily of Edges and Nodes. For very simple designs there is a one-to-one relation between HarnessNode and Vertex on one hand, and HarnessSegment and Edge on the other hand. However for a bit more complex case the need to work with several related harness design views is essential. The link between common HarnessNodes and common HarnessSegments of different WiringHarnessAssemblyDesign of the same final Part is established by common Edges and Vertices within a common RepresentationContext.

Cross-section/Onion

Flat versus hierarchical assembly

Connectivity

5.2. Examples: simple electrical PartView

Example taken from HarnessExample_Hierarchical.stpx: A *Part/PartVersion* that is categorized to be a "**terminal lug**" and "discrete" with a single PartVersion and PartView. The PartView has an initial ViewContext with the ApplicationDomain "electrical" and LifeCycleStage "design". There are two *SingleOccurrences*, "lug1" and "lug2" defined for the PartView. See clause 8 for using these occurrences in an assembly.

```
<ViewContext uid="_100102">
  <ApplicationDomain>
    <PredefinedApplicationDomainEnum>electrical</PredefinedApplicationDomainEnum>
  </ApplicationDomain>
  <LifeCycleStage>
    <ProxyString>design</ProxyString>
  </LifeCycleStage>
</ViewContext>
...
<Part uid="_103000">
  <Id id="MS5036-153"/>
  <Name>
    <LocalizedString lang="en-US">TERMINAL LUG CRIMP STYLE COPPER INSULATED RING
TONGUE</LocalizedString>
    <LocalizedString lang="fr-FR">COSSE</LocalizedString>
  </Name>
  <PartTypes>
    <PartCategoryEnum>discrete</PartCategoryEnum>
    <PartCategoryEnum>terminal_lug</PartCategoryEnum>
  </PartTypes>
  <Versions>
    <PartVersion uid="_103001">
      <Id id="Version 1"/>
      <Views>
        <PartView uid="_103002">
          <DefiningGeometry uidRef="_103090"/>
          <InitialContext uidRef="_100102"/>
          <Occurrence xsi:type="n0:SingleOccurrence" uid="_203005">
            <Id id="lug1"/>
            ...
          </Occurrence>
          <Occurrence xsi:type="n0:SingleOccurrence" uid="_203105">
            <Id id="lug2"/>
            ...
          </Occurrence>
          ...
        </PartView>
      </Views>
    </PartVersion>
  </Versions>
</Part>
```

5.3. Example: simple WiringHarnessAssemblyDesign

Example taken from HarnessExample_Hierarchical.stpx:
A *Part* with the categories “wiring_harness” and “discrete” is defined. Its single *PartVersion* contains a *PartView* of type *WiringHarnessAssemblyDesign*. As all other PartViews used for design of an electrical wire harness this has an initial ViewContext with the ApplicationDomain “electrical” and LifeCycleStage “design”. There are two additional ViewContexts defined for this design view. By ViewContexts "_100104" with the ApplicationDomain “wiring_harness_segment_topology” it is made clear that the *WiringHarnessAssemblyDesign* contains a complete topological representation. By ViewContexts "_100105" with the ApplicationDomain “wire_and_part_list_with_connectivity” it is made clear that the *WiringHarnessAssemblyDesign* contains all electrical relevant occurrences such as wires, cables, connectors, contacts, terminal lugs and more.

```
<ViewContext uid="_100104">
  <ApplicationDomain>
    <PredefinedApplicationDomainEnum>wiring_harness_segment_topology</
PredefinedApplicationDomainEnum>
  </ApplicationDomain>
  <LifeCycleStage>
    <ProxyString>design</ProxyString>
  </LifeCycleStage>
</ViewContext>

<ViewContext uid="_100105">
  <ApplicationDomain>
    <PredefinedApplicationDomainEnum>wire_and_part_list_with_connectivity</
PredefinedApplicationDomainEnum>
  </ApplicationDomain>
  <LifeCycleStage>
    <ProxyString>design</ProxyString>
  </LifeCycleStage>
</ViewContext>

<Part uid="_311000">
  <Id id="Part_H1"/>
  <Name> <CharacterString>Electrical Harness example 1</CharacterString> </Name>
  <PartTypes>
    <PartCategoryEnum>wiring_harness</PartCategoryEnum>
    <PartCategoryEnum>discrete</PartCategoryEnum>
  </PartTypes>
  <Versions>
    <PartVersion uid="_311001">
      <Id></Id>
      <Views>
        <PartView xsi:type="n0:WiringHarnessAssemblyDesign" uid="_311002">"
uid="_311002">
          <AdditionalContexts>
            <ViewContext uidRef="_100104"/>
            <ViewContext uidRef="_100105"/>
          </AdditionalContexts>
          <DefiningGeometry uidRef="_314090"/>
          <InitialContext uidRef="_100102"/>
          ...
          <Topology uidRef="_321010" />
        </PartView>
      </Views>
    </PartVersion>
  </Versions>
</Part>
```

For the attributes DefiningGeometry and Topology of a WiringHarnessAssemblyDesign see later clauses.

5.4. Example: complete and partial WiringHarnessAssemblyDesign

example taken from file: HarnessExample_DesignSplitting.xml

xxx

5.5. Parts used in the reference example

Table 1 lists all the Part used in the reference example, together with the used PartVersion, PartView and part category.

Part	Version	View	Id	Name	Category/type
_101000	_101001	_101002	04034-22-9	WIRE,ELEC,COMP,SNGL CONDUCTOR,150 DEG C	wire, raw material by length
_102000	_102001	_102002	RG 58	RG 58	cable, raw material by length
_103000	_103001	_103002	MS5036-153	TERMINAL LUG CRIMP STYLE COPPER INSULATED RING TONGUE	terminal lug
_104000	_104001	_104002	16 AWG	Speaker wire	cable, raw material by length
_110000	_110001	_110002	SB6 4 1 M G 05 W2 P E1 01 AA	ARINC 600 set	connector kit
_111000	_111001	_111002	8660-31A-100-01A/AA	ARINC 600 shell 1 A and B dummy	connector insert
_112000	_112001	_112002	8660-5W2	ARINC 600 shell 1 C 5W2 insert	connector insert
_113000	_113001	_113002	8660-2485	#5 Coax contact	connector contact
_114000	_114002	_103090	8660-249	#16 Rack plug power contact	connector contact
_115000	_115001	_115002	8660-5W2x1	Insert 5W2 assembly	connector insert
_116000	_116001	_116002	8660-250	#12 Rack plug signal contact	connector contact
_117000	_117001	_117002	TM2PB	Phone connector 6.35mm	connector
_118000	_118001	_118002	M81824	Butt Splice Terminal	splice
_119000	_119001	_119002	DEP09S065TLF	D-sub 9 Pin Db9 Female Solder Type Socket Connector	connector
_120000	_120001	_120002	TC122	Braid 1/2inch	overbraid
_121000	_121001	_121002	F6W1.50BK	Wrap	protective covering
_122000	_122001	_122002	H4N1.00BK	HeatShrink	protective covering
_123000	_123001	_123002	8660-140	Backshell EMI connector	electrified backshell
_311000	_311001	_311002	Part_H1	Electrical Harness team reference example	wiring harness
_411000	_411001	_411002	Part_H2	Electrical Harness example 2 (minmal)	wiring harness
_511000	_511001	_511002	Aircraft99x	Aircraft99x	

Table 4: Parts used in the example

6. ProductConcept/Class and ProductConfiguration

A similar concept to the combination of *Part*, *PartVersion* and *PartView* is available with the combination of *ProductClass* and *ProductConfiguration*. The main difference is that a *Part* has a part-Number (=id) by which it is identified. Also a *PartView* is expected to provide all needed details for a given *ApplicationDomain* (e.g. “electrical”) and *LiveCycleStage* (e.g. “design”). *ProductConcept* together with *ProductConfiguration* on the other hand provides only a few characteristics needed for market presentation. There are no design details. However for variant products *Specifications* can be used to characterize the high level (alternative) options that are available for a *ProductClass*.

Recommendation: Use subtype *ProductClass* always. On the MIM/IR level this is required to have any property values defined for this.

There are two main usages for the use of *ProductClass* & *ProductConfiguration*, one is for the usage of standard piece parts for which only **catalogue data** is available, but no further detailed information. The other use is for the configuration of **high end products** that may come in many different configurations. This usage comes together with the concept of Product *Specifications* that are not covered in this tutorial.

While *ProductClass* & *ProductConfiguration* represents a **customer view**, there might be one or several corresponding Part/PartVersion and PartView to represent detailed **design view**. A *ProductDesignAssociation* links between the customer view and the design view.

A single *Part*/*PartVersion* and *PartView* might be used to capture the design information of several similar *ProductConfigurations* (typically of the same *ProductClass*). For this case the *PartView* (typically subtype *AssemblyDefinition*) has to contain all the needed alternative and optional assembly components and *ShapeFeatures*. So the design contains more than what can actually be manufactured for a single individual. Such a design is called a 150% product structure. *ConfiguredAssemblyEffectivity* is used to select the 100% out of the 150% of the product structure that applies for the associated *ProductConfiguration*.

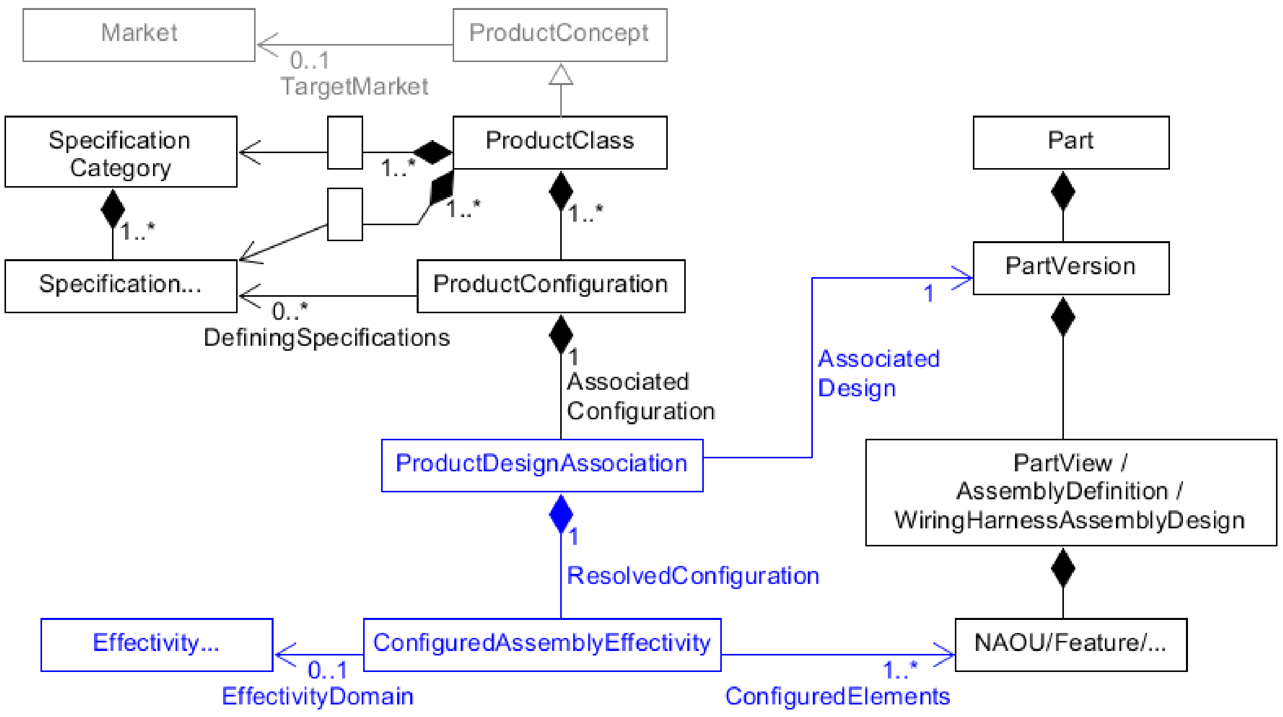


Figure 7: Simplified data model for Product Specification and Configuration

Both, *ProductConfiguration* and *PartView* can be used as *Definition* for *DefinitionBasedOccurrences* that may show up in a WiringHarnessAssemblyDesign. To support this both, *ProductConfiguration* and *PartView* can have *PropertyValueAssignments* (e.g. "open-circuit voltage" = 12V) and *PartTerminals* (e.g. "PLUS" and "MINUS" for the provided battery example).

6.1. Standard piece parts

In this example a standard battery of 12V and a capacity of 100Ah is defined. No design details are provided. For further details such as classification and properties see clause 7. A SingleOccurrence with id “battery1” is defined for this kind of battery that can then be used in an assembly (see clause 8).

Example taken of HarnessExample_Hierachical.xml:

```
<ProductConcept xsi:type="n0:ProductClass" uid="_124000">
  ...
  <Id id="Battery-Std"/>
  <Name>
    <CharacterString>Standard Battery</CharacterString>
  </Name>
  <ProductConfiguration uid="_124001">
    <Id id="Battery-Std"/>
    <Name>
      <CharacterString>Battery,12V,100Ah</CharacterString>
    </Name>
    <Occurrence xsi:type="n0:SingleOccurrence" uid="_224100">
      <Id id="battery1"/>
    </Occurrence>
     ...
  </ProductConfiguration>
</ProductConcept>
```

6.2. Top level assembly with 150% product structure

Example taken out of HarnessExample_Hierachical.xml:
A *ProductClass* for the harness H1 is defined that contains two alternative *ProductConfigurations*, one with a stereo plug and the other with 2 hookup lugs. Both *ProductConfigurations* are associated to the same *WiringHarnessAssemblyDesign* "_311001" (sub-subtype of *PartView*).

The first *ProductConfiguration* disables the occurrences for lug3 and lug4 as well as the corresponding *AssemblyShapeJoints* for these 2 lugs. In the second *ProductConfiguration* the phone1 connector is disabled together with the corresponding *AssemblyShapeJoints* (subtypes of *ShapeElement*) that connects the phone1 terminals.

```
<ProductConcept xsi:type="n0:ProductClass" uid="_390000">
  <Id id="H1 class"/>
  <Name>
    <CharacterString>H1 harness</CharacterString>
  </Name>

  <ProductConfiguration uid="_391000">
    <Id id="H1 stereo plug"/>
    <Name><CharacterString>H1 harness with stereo plug</CharacterString></Name>
    <ProductDesignAssociation uid="_391010">
      <AssociatedDesign uidRef="_311001"/>
      <ConfiguredAssemblyEffectivity uid="_391011">
        <ConfiguredElements>
          <Occurrence uidRef="_203205"/> <!--no lug3-->
          <PartShapeElement uidRef="_311053"/> <!--no lug3/crimp join-->
          <Occurrence uidRef="_203305"/> <!--no lug4-->
          <PartShapeElement uidRef="_311054"/> <!--no lug4/crimp join-->
        </ConfiguredElements>
      </ConfiguredAssemblyEffectivity>
    </ProductDesignAssociation>
  </ProductConfiguration>

  <ProductConfiguration uid="_392000">
    <Id id="H1 hookup lugs"/>
    <Name><CharacterString>H1 harness with hookup lugs</CharacterString></Name>
    <ProductDesignAssociation uid="_392010">
      <AssociatedDesign uidRef="_311001"/>
      <ConfiguredAssemblyEffectivity uid="_392011">
        <ConfiguredElements>
          <Occurrence uidRef="_217100"/> <!--no phone1-->
          <PartShapeElement uidRef="_311020"/> <!--no phone1#Join signal-->
          <PartShapeElement uidRef="_311021"/> <!--no phone1#Join gnd-->
        </ConfiguredElements>
      </ConfiguredAssemblyEffectivity>
    </ProductDesignAssociation>
  </ProductConfiguration>

</ProductConcept>
```

6.3. Highly configurable end product with Specifications and ProductBreakdown

For a wiring harness assembly design with very many variants it might be more useful to use a *Breakdown* structure. AP242 provides this capability, however as of today no example has been worked out for a wiring harness.

7. Classes and Properties

Like many other object types, also *Parts* and *ProductConcepts* can be classified in various ways. AP242 distinguishes between two major kinds of additional information for parts that is category and classification - and they may partly overlap.

PartCategories are used to support the import of electrical wiring harness data in computer systems by defining major kinds of products such as cables, wires, contacts, connectors etc. Only with this upfront information a tool is able to handle electrical wiring product structures in a useful way. For this kind of information see clause 4.2 "*Part* and *ProductClass* Categories specific for Electrical Harness and other electrical applications" for details.

Classification is complementary to *PartCategories* in that it provides linking into dictionaries of more detailed product taxonomies and kinds of predefined properties for these products. In particular dictionaries that are organized according to the Parts Library standards (PLIB) ISO 13584-42 are supported; especially the IEC 61360 - Common Data Dictionary that is relevant for the purpose of electrical wire harness.

For specific classes of products also specific properties are defined in various standards. See clause "6.2 Template 'PropertyAssignment'" in the "Recommended Practices for AP242 Business Object Model XML Assembly Structure" on how to use the DO PropertyValueAssignment, PropertyValue and PropertyDefinition in general.

7.1. Classes and Properties defined in the IEC 61360 - Common Data Dictionary

For the purpose of EWH we have the need to refer into predefined properties in the "IEC 61360 - Common Data Dictionary", see
https://cdd.iec.ch/cdd/iec61360/iec61360.nsf/Welcome?OpenPage

In this dictionary we can e.g. find the

- class "AA017 - battery" with the id "0112/2///61360_4#AAA017#001"
  following the path from "AAA001 - component" and "AAA002 - electric/ electronic component".

Many attributes are defined for the battery class, e.g.

- property "AAE529 open-circuit voltage" with the ID "0112/2///61360_4#AAE529#001"

This has to be used as follows:

1) First we have to identity the IEC organization:

```
<Organization uid="_100430">
  <Id id="IEC"/>
  <Name>
    <CharacterString>International Electrotechnical Commission</CharacterString>
  </Name>
</Organization>
```

2) Next we identity the IEC 61360 dictionary as an ExternalClassSystem:

```
<ExternalClassSystem uid="_100502">
  <Id>
    <Identifier uid="_100503"
      id="IEC 61360 - Common Data Dictionary"
      idRoleRef="_100504"
      idContextRef="_100430" />
  </Id>
</ExternalClassSystem>
```

3) Provide a reference for the battery class within the IEC dictionary with the normative id

"0112/2///61360_4#AAA017".

```
<Class uid="_100501">
  <DefinedIn uidRef="_100502"/>
  <Description>
    <Descriptor uid="_100507">
      <Text>
        <CharacterString>battery</CharacterString>
      </Text>
    </Descriptor>
  </Description>
  <Id id="0112/2///61360_4#AAA017"/>
</Class>
```

4) Provide a reference for the "open-circuit voltage" property defined in the IEC dictionary with the normative ID "0112/2///61360_4#AAE023".

```
<PropertyDefinition uid="_100207">
  <DefinedIn uidRef="_100502"/>
  <Description>
    <Descriptor uid="_100208">
      <Text>
        <CharacterString>open-circuit voltage</CharacterString>
      </Text>
    </Descriptor>
  </Description>
  <Id id="0112/2///61360_4#AAE529"/>
</PropertyDefinition>
```

5) Define the unit “volt” that is a unit for the quantity “voltage” as defined in
IEC 80000-6 "Quantities and units – Part 6: Electromagnetism".

```
<Unit uid="_100302">
  <Name><ClassString>volt</ClassString></Name>
  <Quantity><ClassString>voltage</ClassString></Quantity>
</Unit>
```

6) The most easy way would be to define a new part for the battery and use it, e.g. like this:

```
<Part uid="_124000">
  <Id id="Battery-Std"/>
  <Name> <CharacterString>Battery,12V,100Ah</CharacterString> </Name>
  <PartTypes>
    <PartCategoryEnum>discrete</PartCategoryEnum>
  </PartTypes>
  <Versions>
    <PartVersion uid="_124001">
      <Id></Id>
      <Views>
        <PartView uid="_124002">
          ...
          <InitialContext uidRef="_100102"/>
          ...
          <PropertyValueAssignment uid="_124002">
            <AssignedPropertyValues>
              <PropertyValue xsi:type="n0:NumericalValue" uid="_124012">
                <Definition>
                  <PropertyDefinition uidRef="_100207"/>
                </Definition>
                <Unit uidRef="_100302"/>
                <ValueComponent>12</ValueComponent>
              </PropertyValue>
            </AssignedPropertyValues>
          </PropertyValueAssignment>
        </PartView>
      </Views>
    </PartVersion>
  </Versions>
</Part>
```

7) ... but the above approach would not be fully correct at this battery is not fully specified by a real part number that one could buy or build. Instead this is only a vaguely characterized product for which many different parts are offered on the market. The correct approach is to handle this as a *ProductConcept* subtype *ProductClass* together with a *ProductConfiguration* with the specific open-circuit voltage of 12 volt that is needed:

```
<ProductConcept xsi:type="n0:ProductClass" uid="_124000">
  <ClassifiedAs>
    <Classification uidRef="_100505"/>
  </ClassifiedAs>
  <Id id="Battery-Std"/>
  <Name>
    <CharacterString>Standard Battery</CharacterString>
  </Name>
  <ProductConfiguration uid="_124001">
    <Id id="Battery-Std"/>
    <Name>
      <CharacterString>Battery,12V,100Ah</CharacterString>
    </Name>
    ...
    <PropertyValueAssignment uid="_124002">
      <AssignedPropertyValues>
        <PropertyValue xsi:type="n0:NumericalValue" uid="_124012">
          <Definition>
            <PropertyDefinition uidRef="_100207"/>
          </Definition>
          <Unit uidRef="_100302"/>
          <ValueComponent>12</ValueComponent>
        </PropertyValue>
      </AssignedPropertyValues>
    </PropertyValueAssignment>
  </ProductConfiguration>
</ProductConcept>
```

8. Occurrences and Multi-level Assembly Structure

AP242 allows to build up single and multi level assembly structures. Each assembly, either a top assembly or an intermediate sub-assembly, is represented by an *AssemblyDefinition* that is a subtype of *PartView*. The "components" of an *AssemblyDefinition* are subtypes of *Occurrence* that are related into the assembly through *AssemblyOccurrenceRelationships*. Its supertype *ViewOccurrenceRelationship* allows in a generic way to relate an *Occurrence* to a *PartView.*

In cases that a set of Occurrences belong together but are not physically combined with each other a *CollectionDefinition* is to be used, that is also a subtype of PartView.

An AssemblyDefinition is the relating target of an AssemblyOccurrenceRelationship that is a subtype of ViewOccurrenceRelationship that in a generic way establishes a relation between an Occurrence and a PartView. An AssemblyOccurrenceRelationship is either a NextAssemblyOccurrenceUsage or a PromissoryAssemblyOccurrenceUsage. A NextAssemblyOccurrenceUsage makes an Occurrence a direct constituent of an AssemblyDefinition. A PromissoryAssemblyOccurrenceUsage only indicates that an Occurrence may become an constituent in some way, maybe in some indirect way.

An Occurrence may be defined in three different ways:

1. In most cases an Occurrence is a particular usage of a PartView that inherits all its attributes including the geometric models from the PartView unless it is said otherwise.
2. But an Occurrence may also be a usage of a ProductConfiguration that is a member of a ProductClass. This case is used especially for standard parts that are bought off the shelf with only limited design information.
3. A third case is when an Occurrence is defined by a lower level Occurrence in the context of a higher level Occurrence. The specific subtype SpecifiedOccurrence is used in this case, see below and clause 5.6.

Occurrence itself is an abstract concept. One of the following subtype has to be used:

- A DefinitionBasedOccurrence is an Occurrence that is also abstract and that has as its definition either a PartView or a ProductConfiguration (but not another Occurrence).
  - A SingleOccurrence is a DefinitionBasedOccurrence that is taken from its definition as a single piece.
  - A QuantifiedOccurrence is a DefinitionBasedOccurrence that is taken from its definition in some quantity. The quantity can be defined in the number of pieces (e.g. 10 rivets), or by length (e.g. 2m wire), or by surface (e.g. 3 sqm. of glass) or by volume (2.5 litre of oil). It the quantity can not be specified exactly an alternative criterion can be specified (e.g. enough glue for the whole surface). At least either the quantity or the criterion must be specified.
- A SpecifiedOccurrence is an Occurrence that has as its definition another Occurrence. This definition is used by a NextAssemblyUsage in an AssemblyDefinition. In parallel the SpecifiedOccurrence calls out by the attribute UpperUsage one more other Occurrence, that has as its Definition (directly or indirectly) this AssemblyDefinition.

A single or multi level assembly structure shall form a directed acyclic graph. *ProductConfiguration*s, *PartView*s, *AssemblyDefinition*s and *Occurrence*s are the nodes, while *AssemblyOccurrenceRelationship*s, and the attribute *Occurrence.Definition* are the links between the nodes in this graph. Other than that this graph shall be "directed" and "acyclic" the following constraints apply:

- An Occurrence can be related at most once into an AssemblyDefinition, but an Occurrence can be related several times into different AssemblyDefinitions; e.g. to represent variants.
- In the case that an Occurrence is defined by a PartView, this PartView must not be the AssemblyDefinition into which the Occurrence is related to nor any of its higher AssemblyDefinitions.
- PartViews or AssemblyDefinitions at different levels of an assembly structure shall not share the

same PartVersion nor the same Part.

See the “Recommended Practices for AP242 Business Object Model XML Assembly Structure” for detailed information on how to implement assembly structures in general.

Clause 5.6 provides a detailed example of a hierarchical assembly structure with SpecifiedOccurrence.

Attention: Do not use the types AssemblyViewRelationship, NextAssemblyViewUsage and PromissoryAssemblyViewUsage for electrical harness. These types had been introduced in AP242 for the harmonization with AP239 (PLCS), but they do in no way support the functionality needed for electrical harness.

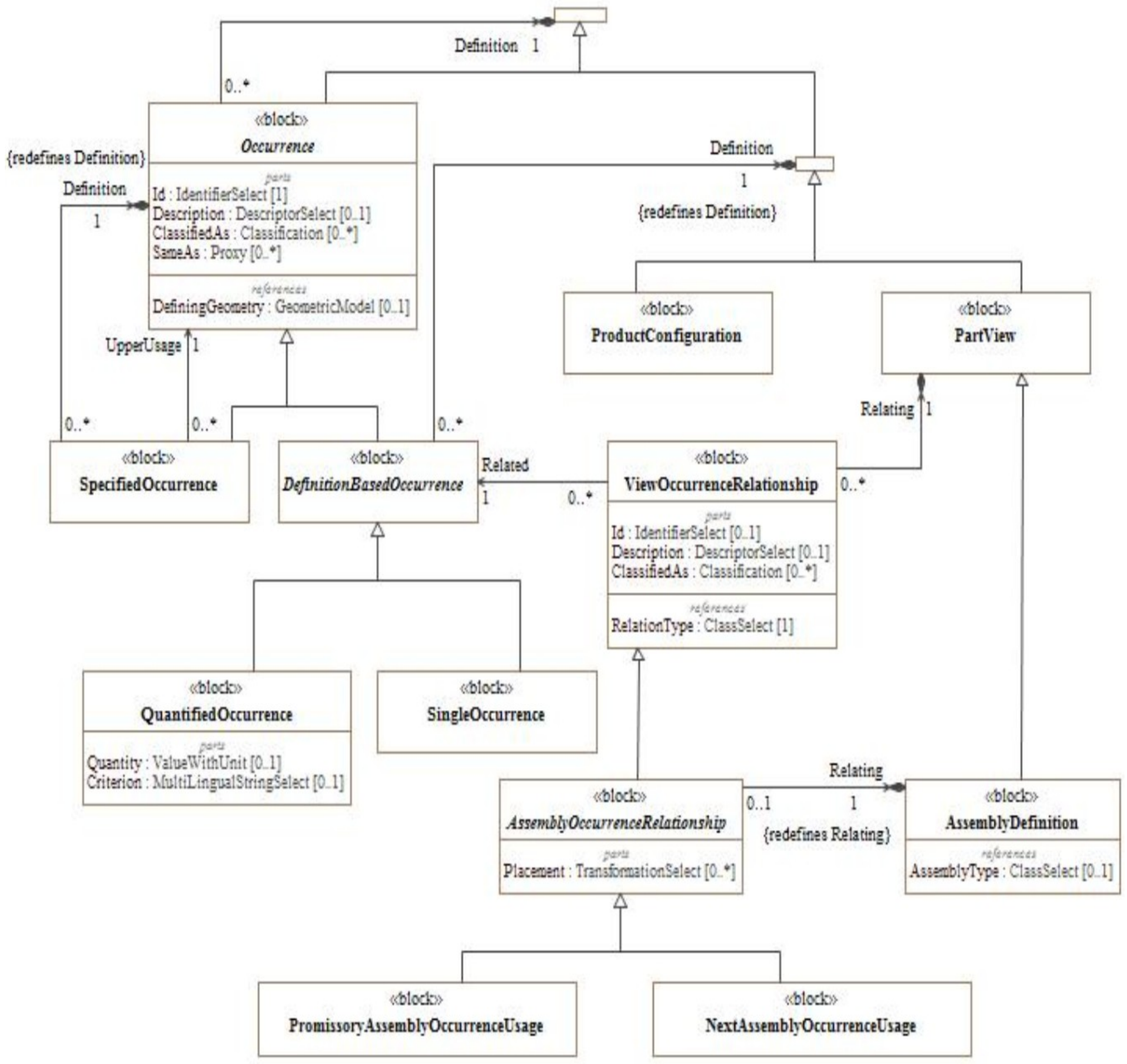

«block»
Occurrence
parts
Id : IdentifierSelect [1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
SameAs : Proxy [0..*]
references
DefiningGeometry : GeometricModel [0..1]
Definition 1
0..*
{redefines Definition}
Definition
1
Definition
1
{redefines Definition}
UpperUsage 1
0..*
0..*
0..*
«block»
ProductConfiguration
«block»
PartView
Relating 1
«block»
SpecifiedOccurrence
«block»
DefinitionBasedOccurrence
Related
1
0..*
«block»
ViewOccurrenceRelationship
parts
Id : IdentifierSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
references
RelationType : ClassSelect [1]
0..*
«block»
QuantifiedOccurrence
parts
Quantity : ValueWithUnit [0..1]
Criterion : MultiLingualStringSelect [0..1]
«block»
SingleOccurrence
«block»
AssemblyOccurrenceRelationship
parts
Placement : TransformationSelect [0..*]
Relating
0..1
1
{redefines Relating}
«block»
AssemblyDefinition
references
AssemblyType : ClassSelect [0..1]
«block»
PromissoryAssemblyOccurrenceUsage
«block»
NextAssemblyOccurrenceUsage

Figure 8: Detailed assembly structure model

The attribute Id of Occurrences shall be unique within all Occurrences used within an AssemblyDefinition. It is used as reference designator. In most cases a simple IdentifierString is sufficient. If there is a need to call out a specific reference designator system (e.g. IEC, IEEE or company specific) the Identifier Object with a Classification can be used.

See an overview of “IEC Reference Designations” at:

http://myelectrical.com/notes/entryid/24/iec-reference-designations

The attribute Id of ViewOccurrenceRelationship and all its subtypes is optional. Recommendation is to not use this attribute.

8.1. Example occurrences

Add Example for SingleOccurrence

Quantified Occurrence

Use only these examples for the next 2 clauses

8.2. Flat assembly

Extract from HarnessExample_flat.xml:

remove wire/cabel ...

H1-example

```
<Part uid="_311000">
  <Id id="Part_H1"/>
  ...
  <Versions>
    <PartVersion uid="_311001">
    <Id/>
    <Views>
      <PartView xsi:type="n0:WiringHarnessAssemblyDesign" uid="_311002">
      ...
        <ViewOccurrenceRelationship uid="_315011"
                 xsi:type="n0:NextAssemblyOccurrenceUsage">
          <Related uidRef="_201004"/>  <!--WireOccurrence wire1-->
          <RelationType><ClassString>next assembly
occurrence</ClassString></RelationType>
                     <Placement>
                        <GeometryToTopologyModelAssociation uidRef="_321080"/>
                     </Placement>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315012"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                     <Related uidRef="_201104"/>
                     <!--WireOccurrence wire2-->
                     <RelationType>
                        <ClassString>next assembly occurrence</ClassString>
                     </RelationType>
                     <Placement>
                        <GeometryToTopologyModelAssociation uidRef="_321081"/>
                     </Placement>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315013"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                     <Related uidRef="_201204"/>
                     <!--WireOccurrence wire3-->
                     <RelationType>
                        <ClassString>next assembly occurrence</ClassString>
                     </RelationType>
                     <Placement>
                        <GeometryToTopologyModelAssociation uidRef="_321082"/>
```

```
                            </Placement>
                        </ViewOccurrenceRelationship>
                        <ViewOccurrenceRelationship uid="_315003"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                            <Related uidRef="_202006"/>
                            <!--CableOccurrence cable1 (RG 58) -->
                            <RelationType>
                                <ClassString>next assembly occurrence</ClassString>
                            </RelationType>
                            <Placement>
                                <GeometryToTopologyModelAssociation uidRef="_321083"/>
                            </Placement>
                        </ViewOccurrenceRelationship>
                        <ViewOccurrenceRelationship uid="_315004"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                            <Related uidRef="_202106"/>
                            <!--CableOccurrence cable2 (RG 58)-->
                            <RelationType>
                                <ClassString>next assembly occurrence</ClassString>
                            </RelationType>
                            <Placement>
                                <GeometryToTopologyModelAssociation uidRef="_321084"/>
                            </Placement>
                        </ViewOccurrenceRelationship>
                        <ViewOccurrenceRelationship uid="_315021"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                            <Related uidRef="_204006"/>
                            <!--CableOccurrence cable3 (speaker wire)-->
                            <RelationType>
                                <ClassString>next assembly occurrence</ClassString>
                            </RelationType>
                            <Placement>
                                <GeometryToTopologyModelAssociation uidRef="_321085"/>
                            </Placement>
                        </ViewOccurrenceRelationship>
                        <ViewOccurrenceRelationship uid="_315031"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                            <Related uidRef="_220005"/>
                            <!-- braid1 -->
                            <RelationType>
                                <ClassString>next assembly occurrence</ClassString>
                            </RelationType>
                            <Placement>
                                <GeometryToTopologyModelAssociation uidRef="_321088"/>
                            </Placement>
                        </ViewOccurrenceRelationship>
                        <ViewOccurrenceRelationship uid="_315032"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                            <Related uidRef="_221005"/>
                            <!-- wrap1 -->
                            <RelationType>
                                <ClassString>next assembly occurrence</ClassString>
                            </RelationType>
                            <Placement>
                                <GeometryToTopologyModelAssociation uidRef="_321086"/>
                            </Placement>
                        </ViewOccurrenceRelationship>
                        <ViewOccurrenceRelationship uid="_315033"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                            <Related uidRef="_222005"/>
```

```
                    <!-- heatshrink1 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                    <Placement>
                      <GeometryToTopologyModelAssociation uidRef="_321087"/>
                    </Placement>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315041"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_203005"/>
                    <!--SingleOccurrence lug1 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                    <Placement>
                      <GeometricRepresentationRelationship uidRef="_314210"/>
                    </Placement>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_1"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_211010"/>
                    <!--arinc1/A-insert -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_2"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_211020"/>
                    <!--arinc1/B-insert -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_3"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_212010"/>
                    <!--arinc1/C-insert -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_4"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_213100"/>
                    <!--arinc1/C-insert/coax1 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_5"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_213200"/>
                    <!--arinc1/C-insert/coax2 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_6"
```

```
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_214100"/>
                    <!--arinc1/C-insert/power3 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_7"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_214200"/>
                    <!--arinc1/C-insert/power4 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315042_8"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_216100"/>
                    <!--arinc1/C-insert/signal5 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315043"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_217100"/>
                    <!-- phone1 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315044"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_218100"/>
                    <!-- splice1 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315045"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_219100"/>
                    <!-- dsub1 -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <ViewOccurrenceRelationship uid="_315046"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                    <Related uidRef="_225100"/>
                    <!-- arinc 1 housing -->
                    <RelationType>
                      <ClassString>next assembly occurrence</ClassString>
                    </RelationType>
                  </ViewOccurrenceRelationship>
                  <Topology uidRef="_321010"/>
                </PartView>
              </Views>
            </PartVersion>
          </Versions>
```

```
      </Part>
```

8.3. Hierarchical Assembly

SO FAR INCOMPLETE

Hierarchy of PartViews and Occurrences:

- piece parts, e.g. contact
- Insert 5W2 assembly
- ARINC 600 assembly
- Part_H1, type WiringHarnessAssemblyDesign
  - SingleOccurrence h1.1
- Aircraft99x
  - no occurrence defined
  - assembly of
    - SingleOccurrence h1.1

Extract from HarnessExample_hierachical.stpx:

```
<Part uid="_411000"> <!-- Part_H2 -->
  <Id id="Part_H2"/>
  <Name>
    <CharacterString>Electrical Harness example 2</CharacterString>
  </Name>
  <PartTypes>
    <PartCategoryEnum>wiring_harness</PartCategoryEnum>
    <PartCategoryEnum>discrete</PartCategoryEnum>
  </PartTypes>
  <Versions>
    <PartVersion uid="_411001">
      <Id></Id>
      <Views>
        <PartView xsi:type="n0:WiringHarnessAssemblyDesign" uid="_411002">
        ...
        <ViewOccurrenceRelationship uid="_415011"
xsi:type="n0:NextAssemblyOccurrenceUsage">
          <Related uidRef="_201304"/> <!--WireOccurrence wire4-->
          <RelationType>
            <ClassString>next assembly occurrence</ClassString>
          </RelationType>
        </ViewOccurrenceRelationship>
        <ViewOccurrenceRelationship uid="_415041"
xsi:type="n0:NextAssemblyOccurrenceUsage">
          <Related uidRef="_203105"/> <!--SingleOccurrence lug2 -->
          <RelationType>
            <ClassString>next assembly occurrence</ClassString>
          </RelationType>
        </ViewOccurrenceRelationship>
<!-- EXPERIMENTAL, waiting for feedback-->
                                             <ViewOccurrenceRelationship uid="_415042"
xsi:type="n0:PromissoryAssemblyOccurrenceUsage">
                                                   <Description>
                                                         <CharacterString>arinc1 will
become available on the top assembly through the H1 harness</CharacterString>
                                                   </Description>
```

```
                                                          <Related uidRef="_210100"/> <!--
SingleOccurrence arinc1 -->
                                                          <RelationType>
                                                            <ClassString>promissory assembly
occurrence</ClassString>
                                                          </RelationType>
                                                   </ViewOccurrenceRelationship>
                                            </PartView>
                                     </Views>
                              </PartVersion>
                       </Versions>
                </Part>

                <Part uid="_511000"> <!-- Aircraft99x -->
                       <Id id="Aircraft99x"/>
                       <Name>
                              <CharacterString>Aircraft99x</CharacterString>
                       </Name>
                       <PartTypes>
                              <PartCategoryEnum>assembly</PartCategoryEnum>
                       </PartTypes>
                       <Versions>
                              <PartVersion uid="_511001">
                                     <Id></Id>
                                     <Views>
                                            <PartView xsi:type="n0:AssemblyDefinition"
uid="_511002">
...

                                                   <ViewOccurrenceRelationship uid="_515021"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                                                          <Related uidRef="_311105"/> <!--
SingleOccurrence h1.1 -->
                                                          <RelationType>
                                                            <ClassString>next assembly
occurrence</ClassString>
                                                          </RelationType>
                                                          <Placement>

       <GeometricRepresentationRelationship uidRef="_314200"/> <!--
TopologyToGeometryAssociationByMultipleElementPairs -->
                                                          </Placement>
                                                   </ViewOccurrenceRelationship>
                                                   <ViewOccurrenceRelationship uid="_515022"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                                                          <Related uidRef="_411105"/> <!--
SingleOccurrence h2.1 -->
                                                          <RelationType>
                                                            <ClassString>next assembly
occurrence</ClassString>
                                                          </RelationType>
                                                   </ViewOccurrenceRelationship>
                                                   <ViewOccurrenceRelationship uid="_515023"
xsi:type="n0:NextAssemblyOccurrenceUsage">
                                                          <Related uidRef="_224100"/> <!--
SingleOccurrence battery1 -->
                                                          <RelationType>
                                                            <ClassString>next assembly
occurrence</ClassString>
                                                          </RelationType>
```

```
                                        </ViewOccurrenceRelationship>
                                    </PartView>
                                </Views>
                            </PartVersion>
                        </Versions>
                    </Part>
```

9. ShapeElements, Features & Beyond

A **ShapeElement** identifies a piece of the shape of some product or of another ShapeElement, but without defining its geometry. Some subtypes of ShapeElements are called ...ShapeFeature. The term "shape feature" in AP242 means that some piece of the outside surface of a product is identified that typically has a specific functional purpose. A shape feature is a "definitional" part of the shape of a product. There are other shapes of a product that are not definitional, e.g. a rotational axis is not part of the outer surface of a product.

A **ShapeFeatureDefinition** is the identification of a feature independent of a particular product. It can be used like a template for several products. This concept is widely used in the electrical world for standardized shapes of e.g. connectors or contacts.

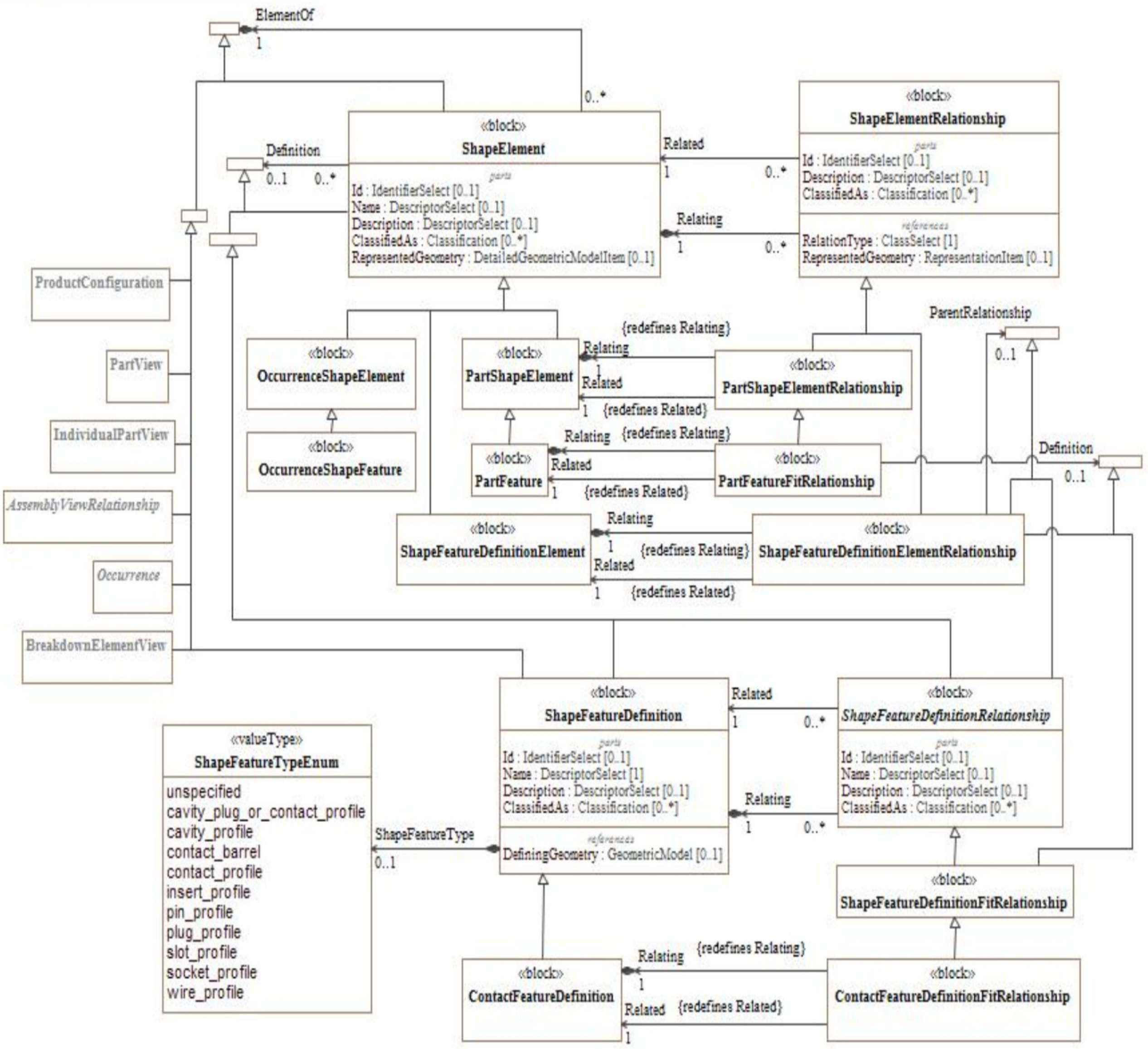

ElementOf
1
0..*
«block»
ShapeElement
parts
Id : IdentifierSelect [0..1]
Name : DescriptorSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
RepresentedGeometry : DetailedGeometricModelItem [0..1]
Definition
0..1
0..*
Related
1
0..*
Relating
1
0..*
«block»
ShapeElementRelationship
parts
Id : IdentifierSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
references
RelationType : ClassSelect [1]
RepresentedGeometry : RepresentationItem [0..1]
ProductConfiguration
PartView
IndividualPartView
AssemblyViewRelationship
Occurrence
BreakdownElementView
«block»
OccurrenceShapeElement
«block»
OccurrenceShapeFeature
«block»
PartShapeElement
{redefines Relating}
Relating
Related
1
{redefines Related}
«block»
PartShapeElementRelationship
ParentRelationship
0..1
«block»
PartFeature
Relating
{redefines Relating}
Related
1
{redefines Related}
«block»
PartFeatureFitRelationship
Definition
0..1
«block»
ShapeFeatureDefinitionElement
Relating
1
{redefines Relating}
Related
1
{redefines Related}
«block»
ShapeFeatureDefinitionElementRelationship
«block»
ShapeFeatureDefinition
parts
Id : IdentifierSelect [0..1]
Name : DescriptorSelect [1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
references
DefiningGeometry : GeometricModel [0..1]
Related
1
0..*
Relating
1
0..*
«block»
ShapeFeatureDefinitionRelationship
parts
Id : IdentifierSelect [0..1]
Name : DescriptorSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
«block»
ShapeFeatureDefinitionFitRelationship
«valueType»
ShapeFeatureTypeEnum
unspecified
cavity_plug_or_contact_profile
cavity_profile
contact_barrel
contact_profile
insert_profile
pin_profile
plug_profile
slot_profile
socket_profile
wire_profile
ShapeFeatureType
0..1
«block»
ContactFeatureDefinition
Relating
{redefines Relating}
1
Related
{redefines Related}
1
«block»
ContactFeatureDefinitionFitRelationship

Figure 9: ShapeElement and ShapeFeatureDefinition

ShapeElements can in principle be defined for ProductConfiguration, PartView, Occurrence, IndividualPartView, AssemblyViewRelationship, BreakdownElementType or ShapeFeatureDefinition. For the purpose of electrical harness only the following specific subtypes of ShapeElements or one of their sub-subtypes are to be used:

- **PartShapeElement** and subtype PartFeature for PartView or ProductConfiguration
- **OccurrenceShapeElement** and subtype OccurrenceShapeFeature for Occurrences

- **ShapeFeatureDefinitionElement** for ShapeFeatureDefinition

ShapeElements can be related in various ways to each other using ShapeElementRelationship. A PartShapeElementRelationship relates two PartShapeElements that might be from the same PartView or from different PartViews. A PartFeatureFitRelationship indicates that two PartFeatures fit together like a plug and a socket. ShapeFeatureDefinitionElementRelationship is providing the same capability on a generic ShapeFeature level independent of a product.

The following pre-defined type information are defined for electrical wiring harness. Whenever suitable these types should be used to enable the proper function of converters:

- cavity_plug_or_contact_profile: an outer profile of a cavity plug or connector contact that fits into a cavity_profile.
- cavity_profile: an inner profile of a connector cavity that can accept a cavity_plug_or_contact_profile.
- contact_barrel: an inner profile of a barrel that can accept a wire_profile.
- contact_profile: an inner profile of a socket contact into which a pin_profile fits.
- insert_profile: an outer profile of a connector insert that fits into a slot_profile.
- pin_profile: an outer profile of a pin contact that fits into a contact_profile.
- plug_profile: an outer profile of a plug that fits into a socket_profile.
- slot_profile: an inner profile of a connector slot that can accept an insert_profile.
- socket_profile: an inner profile of a socket into which a plug_profile fits.
- unspecified: the kind of shape is not specified.
- wire_profile: an outer profile of a wire or cable that fits into a contact_barrel.

SO FAR INCOMPLETE: Main usage in clauses 10.x ff. Explain al concepts of above diagra,

9.1. *Templates for Slots, Inserts, Cavities and Contact Profiles*

In the cases that a particular shape feature is used for several PartDesignViews there is a possibility to define a feature only once, and then to used it for several PartDesignViews by reference.

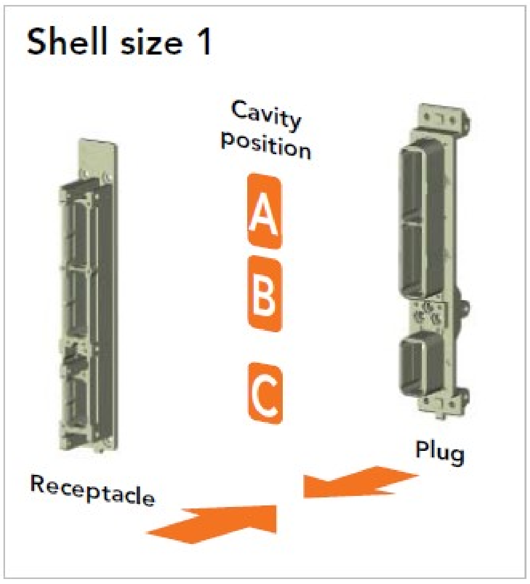
Examples for this are modular connectors such as the ARINC 600 series that are available from different vendors and variations and where different kind of inserts fit into the same slot, or differerent kind of contacts fit into a cavity of a certain size and shape.

xx PartShapeElements defined by ShapeFeatureDefinitions

xx TBD

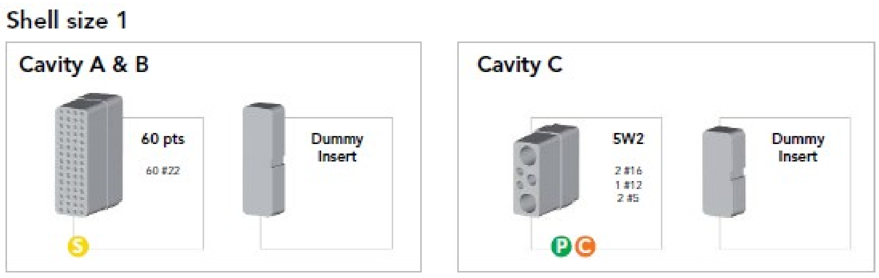
Example pictures for the following pairs:

- insert_profile & slot_profile
- plug_profile and socket_profile
- contact_barrel & wire_profile
- cavity_plug_or_contact_profile & cavity_profile



xx

xx



xx TBD

xx

xx TBD

9.2. *Example for ShapeFeatureDetinitionFitRelationship & PartFeatureFitRelationship*

xxx

10. Electrical Connectivity & AssemblyShapeJoint

The electrical connectivity in AP242 follows the so called "CFI five-box model of electrical connectivity", defined originally by the CAD Framework Initiative (CFI), see
http://www10.edacafe.com/book/ASIC/CH09/CH09.5.php

The CFI model is used in AP242 for both, functional and physical connectivity. As this tutorial focuses on the physical design aspects of an electrical harness, only the physical connectivity aspects are presented here.

All terminal, connectivity and many other geometrically related aspects in AP242 are realised by specializations of the generic ShapeElement and ShapeFeatureDefinition objects.

10.1. Adopted CFI Five-Box Model for Electrical Connectivity

Adopting the CFI five-box model to the modelling style and terminology used in AP242-XML and applying several simplifications the model would looks like this:

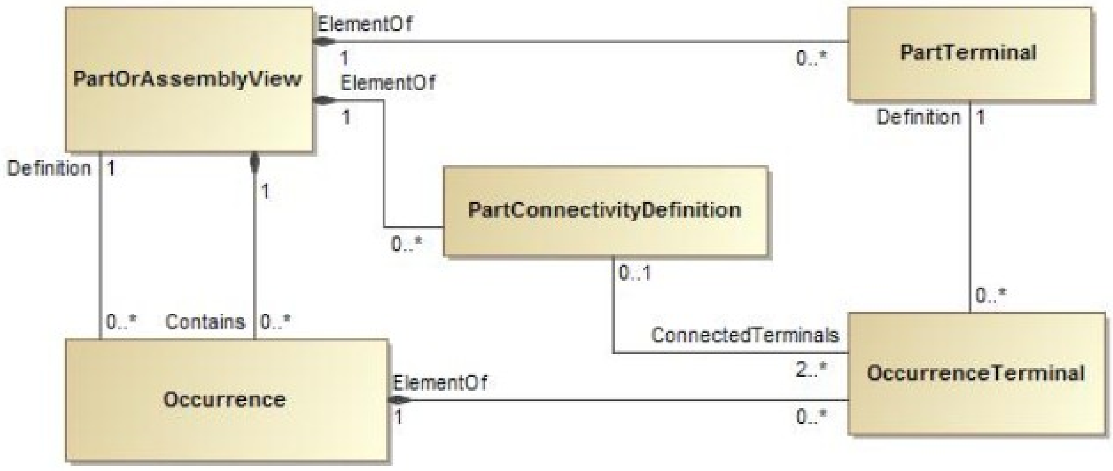


Figure 10: Simplified adopted CFI five-box model for physical connectivity

- A *PartView* represents any kind of a physical design and all the underlying parts it is made of. So the design of an electrical harness is a part and also all the connectors, cables and other things it is made of are represented as *PartView*s.
- A *PartTerminal* represents some area of a *PartView* where it can be electrical or optical connected; e.g. one of the inside or outside terminals of a connector.
  A *PartTerminal* belongs to exactly one *PartView*.
  A *PartView* may have zero, one or many *PartTerminal*s
- An *Occurrence* is a particular usage of a *PartView* and is therefore defined by one *PartView*.
  A *PartView* may serve as the definition for many *Occurrence*s.
  A higher level *PartView* may "contain" *Occurrence*s of lower level *PartView*s In STEP the subtype AssemblyView is to be used in this case. The structure is recursive and so also occurrences of higher level PartView can be constructed that may be contained in an even higher level PartView and so on. PartViews and Occurrences form a Direct Acyclic Graph, so a PartView can not contain Occurrences of itself.
  Example: An electrical harness PartView that contains two Occurrences of a connector.
  Note: This simplified model assumes that an Occurrence is ues in exactly one Part/AssemblyView. In STEP an Occurrence can be used in zero, one or many Part/AssemblyViews.

- An OccurrenceTerminal represents some area of an Occurrence where it can be connected.
  An OccurrenceTerminal belongs to exactly one Occurrence.
  An Occurrence can have zero, one or many OccurrenceTerminals
  An OccurrenceTerminal is defined by one PartTerminal from which it inherits all characteristics.
  A PartTerminal may serve as the definition for many OccurrenceTerminals.
- A PartConnectivityDefinition represents a node of one or several electrical connected elements. It is equivalent to the concept of a Kirchhoff node.
  A PartConnectivityDefinition belongs to one PartView.
  A PartView may have zero, one or many PartConnectivityDefinitions,
  A PartConnectivityDefinition connects two or more PartTerminals or OccurrenceTerminals. In the case of PartTerminals they must be elements of the same PartView. On the case of OccurrenceTerminals they must be elements of Occurrences that are assembly components of the PartView. Consequently the subtype AssemblyView is to be used. So in both cases the connected terminals are in the direct scope of the Part or AssemblyView.
  An OccurrenceTerminal can be used by at most one PhysicalNode for the same PartView.

The following clauses explain in detail how the simplified CFI model is realised in AP242.

10.2. Adopted CFI Five-Box Model for any Joints in an Assembly

AP242 also contains a variation of the CFI five-box model to represent any kind of mechanical connectivity in an assembly (figure 11). This is based on the entities *PartShapeElement* and *OccurrenceShapeElement* that are supertypes of *PartTerminal* respectively *OccurrenceTerminal*. An *AssemblyShapeJoint* is itself a *PartShapeElement* and this belongs to a *PartView* that is an assembly. An *AssemblyShapeJoint* connects typically two or more *OccurrenceShapeElements* through *AssemblyShapeJointItemRelationships*. So an *AssemblyShapeJoint* plays a similar role as a *PartConnectivityDefinition* but is not restricted to ShapeElements that are terminals.

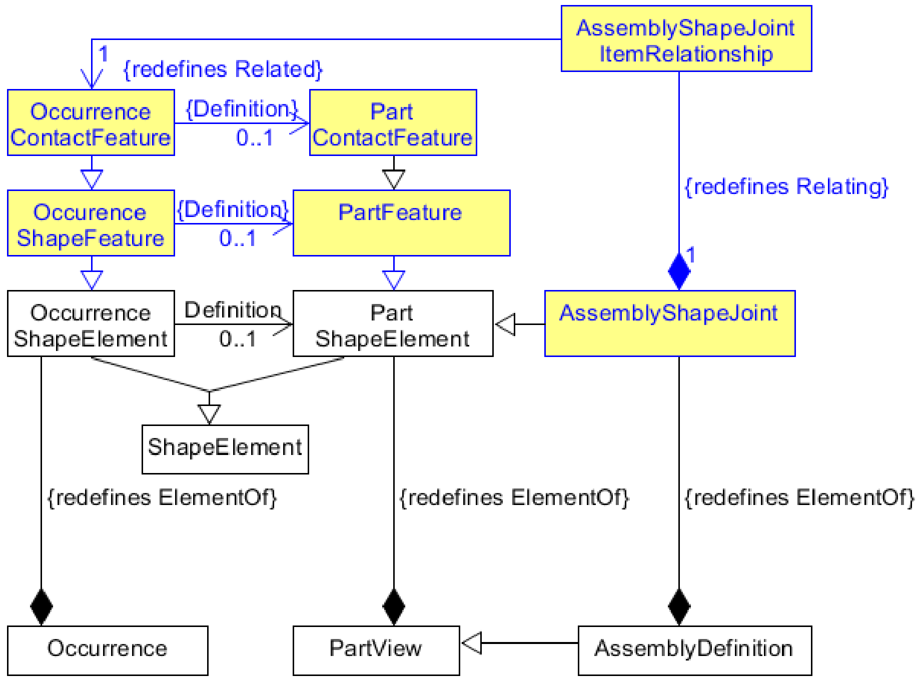


Figure 11: AssemblyShapeJoint, on Occurrence/Part Contact Features

Figure 12 shows an example of the CFI 5 box model for an assembly of two occurrences of a part with 2 terminals.

- top: PartView A with two PartTerminals A1 and A2

- middle: Aa and Ab of PartView A. Each has two OccurrenceTerminals that are defined by the PartTerminals of PartView A.
  Occurrence Aa has the OccurrenceTerminals Aa1 and Aa2.
  Occurrence Ab has the OccurrenceTerminals Ab1 and Ab2.
- bottom: AssemblyView B with two AssemblyShapeJoints.
  AssemblyShapeJoint Bx joints the OccurrenceTerminals Ab1 with Aa2.
  AssemblyShapeJoint By joints the OccurrenceTerminals Ab2 with Aa1.

OccurrenceTerminals are essential to exactly specific which terminals are connected or joined together. The same is true for the more generic supertype OccurrenceShapeElement.

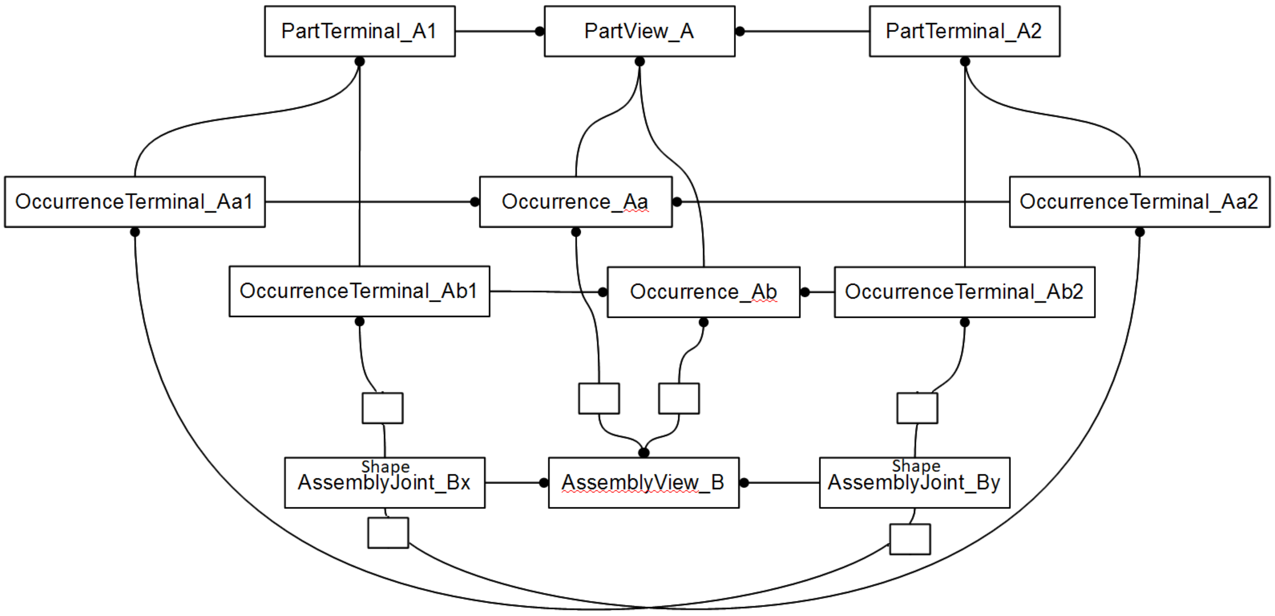


Figure 12: Example of adopted CFI 5 box pattern for AssemblyShapeJoint of terminals

10.3. Detailed ShapeElement and AssemblyShapeJoint Model

This clause explains in detail how two *OccurrenceTerminals* within an assembly are joined together, e.g. an end of a wire with an “internal” terminal of a connector, or two “external” terminals of two connectors.

While in the previous clauses we showed pieces of the AP242 connectivity model in a rather simplified way to highlight the principles, we will now look to it in more detail.

As stated earlier *OccurrenceShapeElement* and *PartShapeElement* are both subtypes of *ShapeElement*. A *PartShapeElement* represents an element of the shape of a *PartView*, while an *OccurrenceShapeElement* represents an element of the shape of an *Occurrence*. To enforce this the *ElementOf* attribute of a *PartShapeElement* must call out a *PartView*, while the *ElementOf* attribute of an *OccurrenceShapeElement* must call out an *Occurrence*.

Note: ElementOf of a ShapeElement might also refer to an AssemblyViewRelationship, BreakdownElementView or IndividualPartView, but these cases are not addressed in this tutorial.

An *OccurrenceShapeElement* calls out as its *Definition* a *PartOrOccurrenceShapeElementSelect* that is either a *PartShapeElement* or an *OccurrenceShapeElement*. There is a constraint on the combination of the values for *ElementOf* and *Definition*. If and only if *ElementOf* has a value of type *SpecifiedOccurrence* the *Definition* attribute must have a value of type *OccurrenceShapeElement*.

An *AssemblyShapeJoint* is a *PartShapeElement* that belongs to an *AssemblyDefinition*. An *AssemblyShapeJoint* connects typically two or more *OccurrenceShapeElements*. In most cases the *OccurrenceShapeElements belong to* different *Occurrences* but there are special cases where some of them may be from the same *Occurrence*. These *Occurrences* must be related into the same *AssemblyDefinition* as the one the *AssemblyShapeJoint* belongs to.

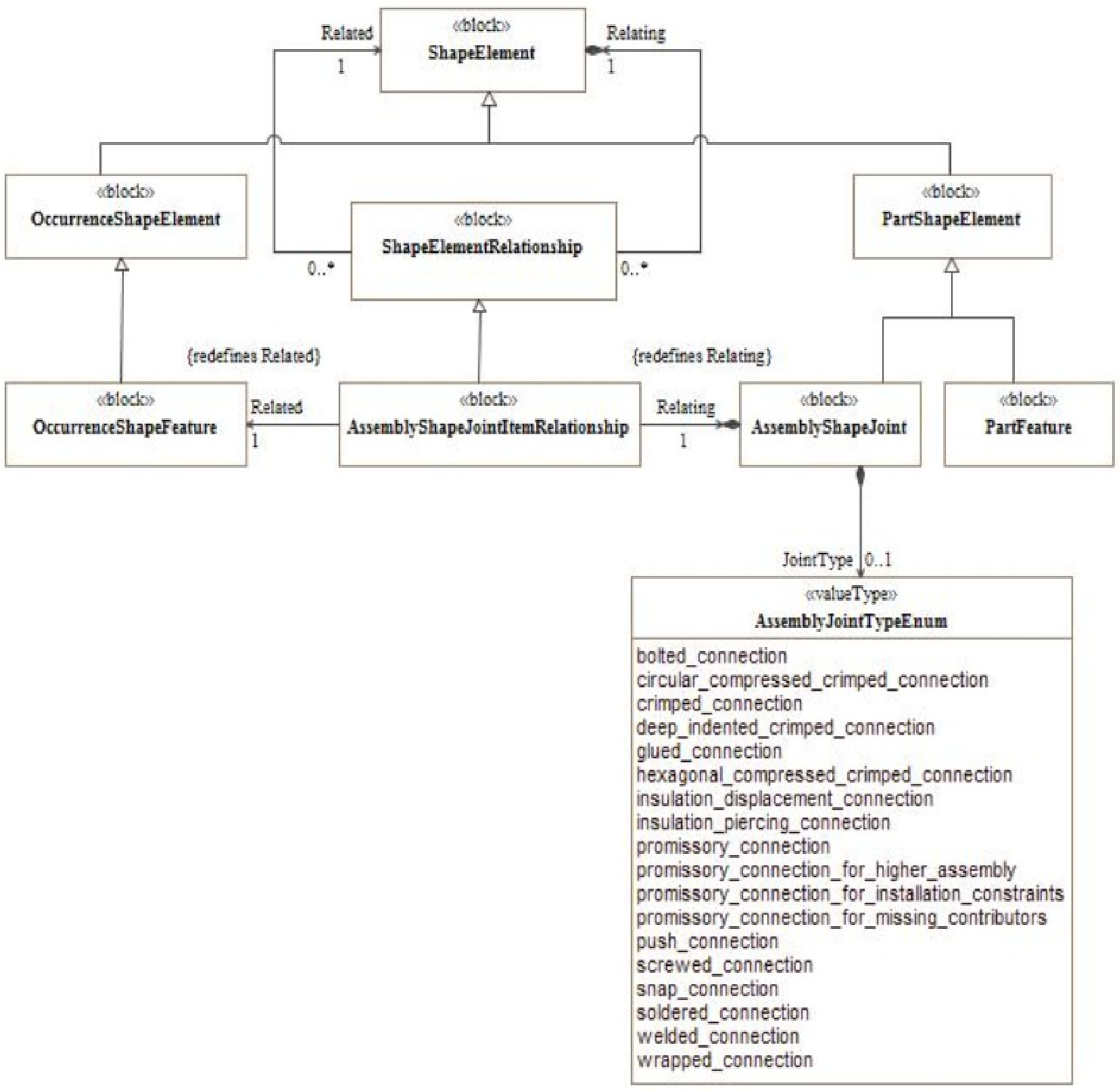

«block» ShapeElement
Related 1
Relating 1
«block» OccurrenceShapeElement
«block» ShapeElementRelationship
0..*
0..*
«block» PartShapeElement
{redefines Related}
{redefines Relating}
«block» OccurrenceShapeFeature
Related 1
«block» AssemblyShapeJointItemRelationship
Relating 1
«block» AssemblyShapeJoint
«block» PartFeature
JointType 0..1
«valueType» AssemblyJointTypeEnum
bolted_connection
circular_compressed_crimped_connection
crimped_connection
deep_indented_crimped_connection
glued_connection
hexagonal_compressed_crimped_connection
insulation_displacement_connection
insulation_piercing_connection
promissory_connection
promissory_connection_for_higher_assembly
promissory_connection_for_installation_constraints
promissory_connection_for_missing_contributors
push_connection
screwed_connection
snap_connection
soldered_connection
welded_connection
wrapped_connection

Figure 13: Detailed AssemblyShapeJoint model

For each *AssemblyShapeJoint* the kind of joint can be specified. The following *JointTypes* are pre-defined:

- **welded connection** [IEV ref 581-23-06]: connection made by welding
- **soldered connection** [IEV ref 581-23-04]: connection made by soldering
- **glued connection**: connection made by glueing
- **snap connection**: connection made by snapping
- **push connection**: connection made by pushing
- **screwed connection**: connection made by screwing
- **bolted connection** [IEV ref 461-19-05]: connection in which the pressure to the conductor is applied by bolting
- **wrapped connection** [IEV ref 581-23-07]: solderless connection achieved by wrapping a solid conductor around a wrap post

- **insulation displacement connection** [IEV ref 581-23-35]: solderless connection made by inserting a single wire into a slot in an insulation displacement termination
- **insulation piercing connection** [IEV ref 461-19-06]: connection made by metallic protrusions which pierce the insulation of the cable core
- **crimped connection** [IEV ref 461-19-01] permanent connection made by the application of pressure inducing the deformation or reshaping of the barrel around the conductor of a cable
  Note – In some cases, the deformation or reshaping of the barrel may affect the form of the conductor.
- **circular compressed crimped connection** [IEV ref 461-19-02]: crimp connection in which the barrel is compressed maintaining essentially its circular form
- **hexagonal compressed crimped connection** [IEV ref 461-19-03]: crimp connection in which the barrel is compressed and reshaped essentially to a hexagonal form
- **deep indented crimped connection** [IEV ref 461-19-04]: crimp connection in which the barrel and the cable conductor are deformed by deep indentations
- **promissory connection**: connection that is promised or expected to be established in some way, but that is by purpose not realised for this PartView
- **promissory connection for higher assembly**: promissory connection that will be realised in a higher assembly in which this assembly is an Occurrence
- **promissory connection for installation constraints**: the connection can not be realised on this assembly level because of installation constraints and this must be realised in a higher assembly
- **promissory connection for missing contributors**: the connection can not be realised on this assembly level because one or more needed OccurrenceShapeElements are not part of this assembly. So the final AssemblyShapeJoint can only be realised in a higher assembly where the other OccurrenceShapeElements become accessible.

The attribute *JointType* is optional; it should only be provided if available in the sending system. Also *JointType* does not need to be specified if one of the underlying *JointTypes* of the affected *PartTerminals* that are called out as *Definition* of the *OccurrenceTerminal* are already specific enough.

However if both, the *JointType* of the *AssemblyShapeJoint* and the JointType of one or more *PartTerminals* are provided, then the *JointTypes* of the *AssemblyShapeJoint* overrides the *JointTypes* of the *PartTerminal(s)*. E.g. if the JointType of the PartTerminal type is a *crimp terminal*, and it used by an *AssemblyShapeJoint* with type *solder*, then this means the joint is realised by soldering, not by crimping.

TBD add example code

AssemblyShapeJoint provides the capability to directly connect two or more terminals. However in some special cases it might be needed to provide a hierarchical structure of how the joints are realized. Example of such a situation with two *AssemblyShapeJoint*s for joining three *OccurrenceTerminal*s (figure 14). First the ends of two wires (wire1 and wire2) are twisted together with a lower level *AssemblyShapeJoint*, and then a higher level *AssemblyShapeJoint* is taking the twised pair and connects it with an *OccurrenceTerminal* from a splice1 by crimping.

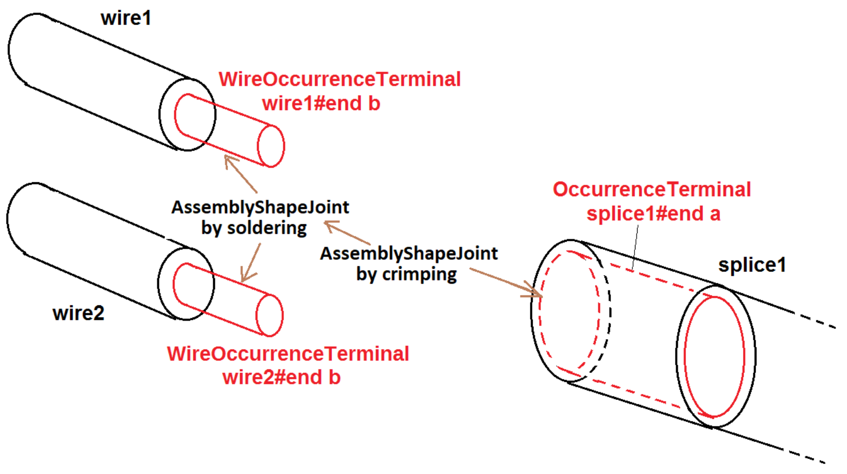


Figure 14: Tree of AssemblyShapeJoints

10.3.1 Example for mechanical AssemblyShapeJoint

xxx Contact in Cavity or Insert in Slot

10.4. Detailed PartTerminal Model

This clause explains in detail the concept of *PartTerminal* that identifies a feature of a part that can be connected with others either electrical or optical.

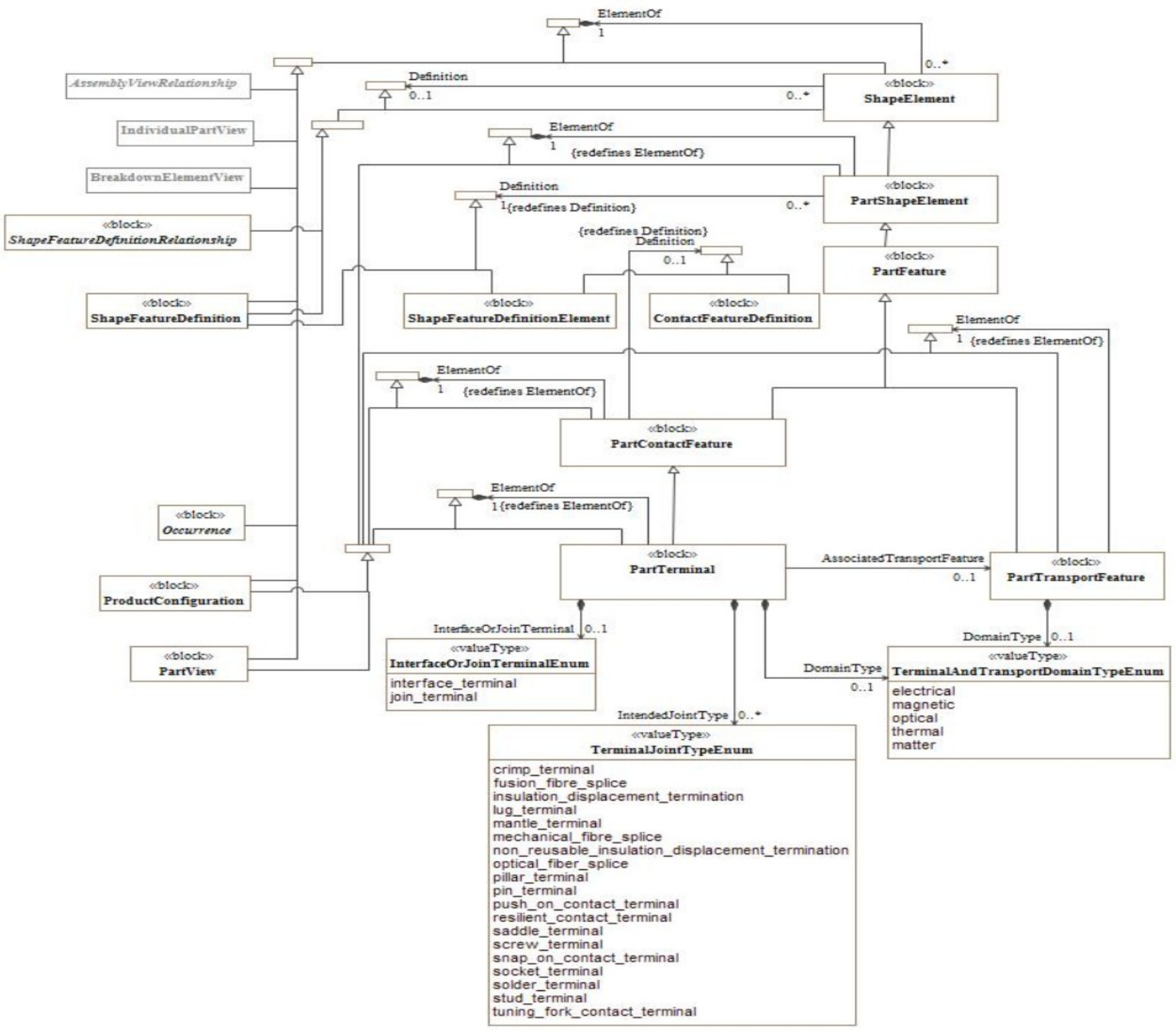


Figure 15: Detailed PartTerminal model

A PartTerminal carries three additional attributes in addition to those it inherits from PartShapeElement:

JointType characterises the intended way on how to contact the PartTerminal. Most of the allowed values listed below are taken from IEC definitions. To achieve a systematic naming convention the word "contact" has been replaced by the word "terminal" or the word "terminal" has been added at the end. The terms written in bold characters shall be used. Alternative IEC terms are separated by comma.

- **socket terminal**, socket contact, female contact [IEV ref 151-12-17]: contact member intended to make electric engagement on its inner surface for mating with the outer surface of another contact member
  Note – In English, the term "socket contact" does not imply that socket contacts are always mounted in a socket nor that sockets have only socket contacts.
- **pin terminal**, pin contact, male contact [IEV ref 151-12-18]: contact member intended to make electric engagement on its outer surface for mating with the inner surface of another contact member
- **crimp terminal**, crimp contact [IEV ref 581-22-05]: contact having a conductor barrel designed to be crimped
- **solder terminal** [IEV ref 442-06-20]: a conductive part of a connecting device provided to enable a termination to be made by means of solder
- **lug terminal** [IEV ref 442-06-16]: a screw-type terminal designed for clamping a cable lug or bar

directly or indirectly by means of a screw or nut

- **screw terminal** [IEV ref 442-06-08]: a terminal, in which the conductor is clamped under the head of one or more screws, and where the clamping pressure can be applied directly by the head of the screw or through an intermediate part, such as a washer, clamping plate or an anti-spread device
- **mantle terminal** [IEV ref 442-06-14]: a terminal, in which the conductor is clamped against the base of a slot in a threaded stud by means of a nut, by a suitably shaped washer under the nut, by a central peg if the nut is a cap nut, or by an equally effective means for transmitting the pressure from the nut to the conductor within the slot
- **saddle terminal** [IEV ref 442-06-09]: a terminal, in which the conductor is clamped against the base of a slot in a threaded stud by means of a nut, by a suitably shaped washer under the nut, by a central peg if the nut is a cap nut, or by an equally effective means for transmitting the pressure from the nut to the conductor within the slot
- **pillar terminal** [IEV ref 442-06-22]: a screw type terminal, in which the conductor(s) is (are) inserted into a hole or cavity, where it is clamped under the shank of the screw
  Note – The clamping pressure can be applied directly by the shank of the screw or through an intermediate part, to which pressure is applied by the shank of the screw.
- **stud terminal** [IEV ref 442-06-23]: a screw-type terminal in which the conductor is clamped under a nut
  Note – The clamping pressure can be applied directly by a suitably shaped nut or through an intermediate part, such as a washer, a clamping plate or an anti-spread device.
- **resilient contact terminal** [IEV ref 581-22-09]: contact having elastic properties to provide a force to its mating part
- **snap on contact terminal** [IEV ref 581-22-10]: push-on contact in which retention is achieved by means of a deformation of the contact area which provides positive axial location
- **tuning fork contact terminal** [IEV ref 581-22-12]: resilient contact having a shape similar to that of a tuning fork, the two arms of which apply contact force in opposite directions
- **push on contact terminal** [IEV ref 581-27-04]: contact with which a connection is achieved by axial force, separation being restricted by friction
- **insulation displacement terminal**, insulation displacement termination [IEV ref 581-23-39]: termination having slots with precisely controlled sides, which are intended to accept a wire and to displace its insulation, deform its conductor and to produce a gas-tight solderless connection
- **non reusable insulation displacement termination** [IEV ref 581-23-41]: insulation displacement termination that can be terminated only once
- **fusion fibre splice**, fusion splice [IEV ref 731-05-06]: a splice accomplished by the application of localised heat sufficient to fuse or melt the ends of two lengths of optical fibre, to produce a continuous single optical fibre
- **mechanical fibre splice**, mechanical splice [IEV ref 731-05-07]: a fibre splice accomplished by fixtures or materials, rather than by thermal fusion
- **optical fibre splice** [IEV ref 731-05-05]: a permanent joint whose purpose is to couple optical power between two optical fibres
  Note – Associated terms : to splice, splicing.

DomainType: Either electrical or optical. In principle also the domain types thermal or magnetic are available, but these are not used in practise so far.

InterfaceOrJoinTerminal: This indicates on whether a PartTerminal is intended to be connected on the next higher assembly level (e.g. the internal terminals of a connector) or whether the PartTerminals are intended to be used to “interface” on an even higher assembly level (e.g. the external terminals of a connector)

10.4.1 Example: Electrical Part with Terminals

xxx TransportFeature with join and interface terminals, e.g. a single contact

10.5. Detailed OccurrenceTerminal and PhysicalNode Model

This clause explains in detail how "components" of an electrical assembly are connected with each other, either by AssemblyShapeJoints or by *PartConnectivityDefinition* that represents nodes in the sense of the Kirchhoff's laws on electrical circuits.

For the exchange of electrical or optical connectivity information, *PartTerminal*, subtype of *PartShapeElement*, and *OccurrenceTerminal*, subtype of *OccurrenceShapeElement*, are to be used. The *Definition* attribute of *OccurrenceTerminal* is of type *PartOrOccurrenceTerminalSelect* which is either a *PartTerminal* or another *OccurrenceTerminal*. Similar to *OccurrenceShapeElement*, the *Definition* of *OccurrenceTerminal* shall only, and only if, be of type *OccurrenceTerminal* when *ElementOf* has a value of type *SpecifiedOccurrence*.

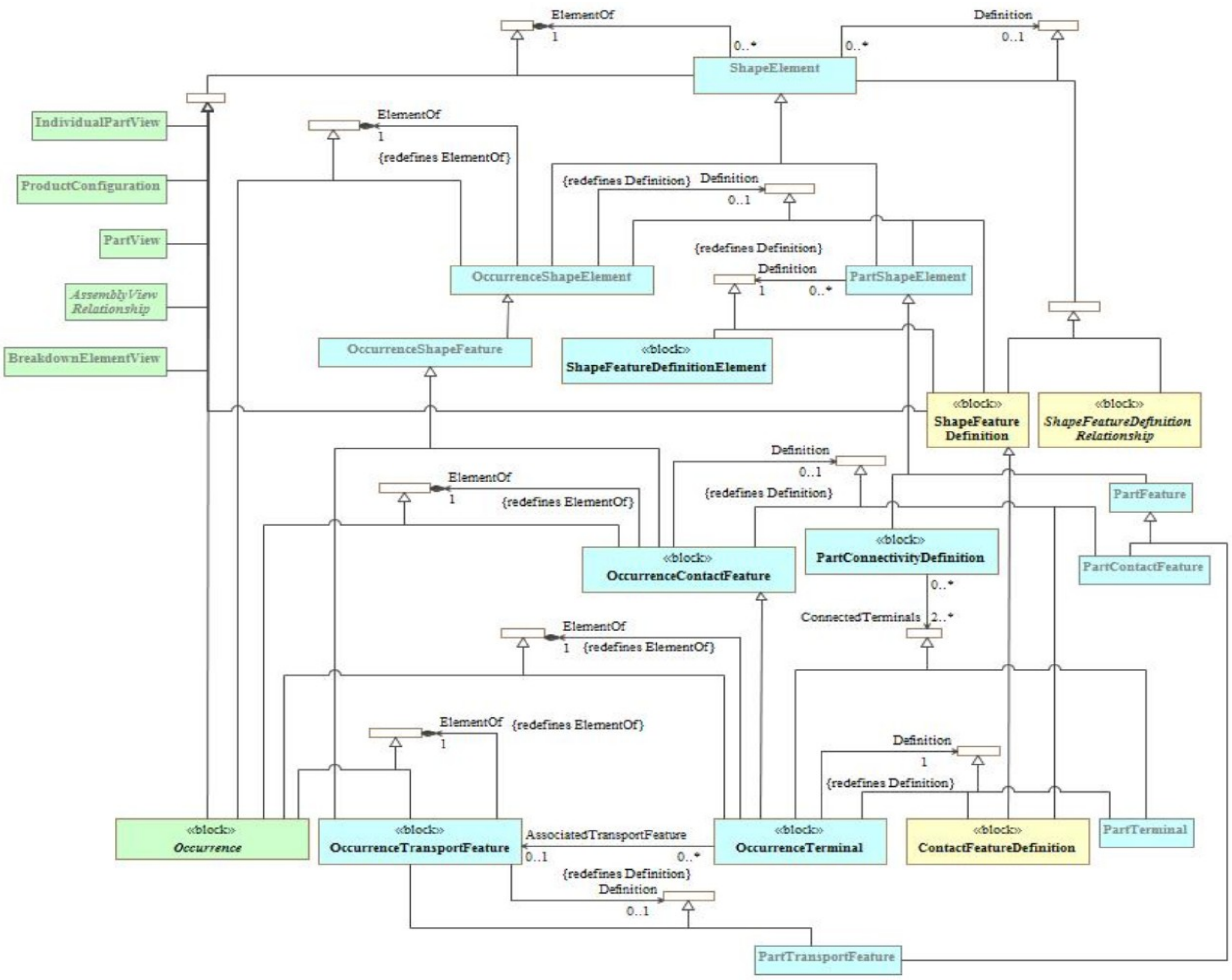


Figure 16: PhysicalNode & OccurrenceTerminal

Like *AssemblyShapeJoint*, *PartConnectivityDefinition*, is also a subtype of *PartShapeElement*. It defines in a more generic way either electrical or optical connection between several terminals without providing all the details. This is done through the attribute *ConnectedTerminals* that is of type *PartOrOccurrenceShapeElement*. The following constraints apply:

- If a member of the *ConnectedTerminals* is of type *PartTerminal*, then this *PartTerminal* shall belong to the same *PartView* to which also *PartConnectivityDefinition* belongs to

- If a member of the *ConnectedTerminals* is of type *OccurrenceTerminal*, then the *Occurrence* this *OccurrenceTerminal* belongs to shall be related into an *AssemblyDefinition* via a *AssemblyOccurrenceRelationship* that is also the same *AssemblyDefinition* the *PhysicalNode* belongs to.

10.5.1 Example: Phone jack occurrence

Example: Pick example with 2 join/interface terminals (e.g. phone jack)

A *PartView* contains two *PartTerminals* whose internal connection is stated by a *PartConnectivityDefinition*.

```
<PartView uid="_103002">
  <DefiningGeometry uidRef="_103090"/>
  <InitialContext uidRef="_100102"/>
  ...
  <ShapeElement xsi:type="n0:PartTerminal" uid="_103003">
    <Id id="External"/>
    <RepresentedGeometry uidRef="_103092"/>
    <IntendedJointType>
      <TerminalJointTypeEnum>screw_terminal</TerminalJointTypeEnum>
    </IntendedJointType>
      <InterfaceOrJoinTerminal>interface_terminal</InterfaceOrJoinTerminal>
  </ShapeElement>
  <ShapeElement xsi:type="n0:PartTerminal" uid="_103004">
    <Id id="Internal"/>
    <RepresentedGeometry uidRef="_103094"/>
    <IntendedJointType>
      <TerminalJointTypeEnum>crimp_terminal</TerminalJointTypeEnum>
    </IntendedJointType>
    <InterfaceOrJoinTerminal>join_terminal</InterfaceOrJoinTerminal>
  </ShapeElement>
  <ShapeElement xsi:type="n0:PartConnectivityDefinition" uid="_103007">
    <ConnectedTerminals>
      <PartTerminal uidRef="_103003"/>
      <PartTerminal uidRef="_103004"/>
    </ConnectedTerminals>
  </ShapeElement>
</PartView>
```

10.6. Referencing Terminals in Hierarchical Assembly Structures

So far we explained only on how to use *AssemblyShapeJoints* and *PartConnectivityDefinitions* within a single assembly level as these concepts can refer only to *OccurrenceShapeElements* respectively *OccurrenceTerminals* within the same assembly level. But what to do when the *AssemblyShapeJoints* and *PartConnectivityDefinitions* need to span over a multi level assembly structure?

The answer is to use *SpecifiedOccurrence* to make *OccurrenceShapeElements*/*OccurrenceTerminals* from a lower assembly level visible on a higher assembly level.

A *SpecifiedOccurrence* has as its *Definition* another *Occurrence* (or subtype *SpecifiedOccurrence*) and calls out with its attribute *UpperUsage* a higher level *Occurrence*, and thus making the lower level *Occurrence* visible in the higher level *Occurrence*. In a similar way the *OccurrenceShapeElements* and *OccurrenceTerminals* from the lower level *Occurrence* can be reflected for the higher level Occurrence by creating copies of *OccurrenceShapeElements* and *OccurrenceTerminals* that refer with their *Definition* attributes to the lower level *OccurrenceShapeElements* and *OccurrenceTerminals*.

Here an example on how this mechanism works through an assembly hierarchy with four levels:

- The top assembly “Aircraft99x” contains an occurrence “h1.1” that is a “Part_H1”
- The assembly “Part_H1” contains an occurrence “arinc1” that is a “ARINC 600 set”

- The assembly “ARINC 600 set” contains an occurrence “C-Assy” for the insert at position C that is a “5W2 assy”
- The assembly “5W2 assy” contains an occurrence “power3” that is a “#16 Rack plug power contact”

A series of SpecifiedOccurrences flatten this hierarchical assembly structure:

- “h1.1/arinc1” represents the “arinc1” occurrence for the higher occurrence “h1.1”
- “arinc1/C-Assy” represents the “A-Assy” occurrence for the higher occurrence “arinc1”
- “h1.1/arinc1/C-Assy” represents the “arinc1/A-Assy” occurrence for the higher occurrence “arinc1/C-Assy”
- “C-Assy/power3” represents the “power3” occurrence for the higher occurrence “C-Assy”
- “arinc1/C-Assy/power3” represents the “C-Assy/power3” occurrence for the higher occurrence “arinc1”
- “h1.1/arinc1/C-Assy/power3” represents the “arinc1/C-Assy/power3” occurrence for the higher occurrence “h1.1”

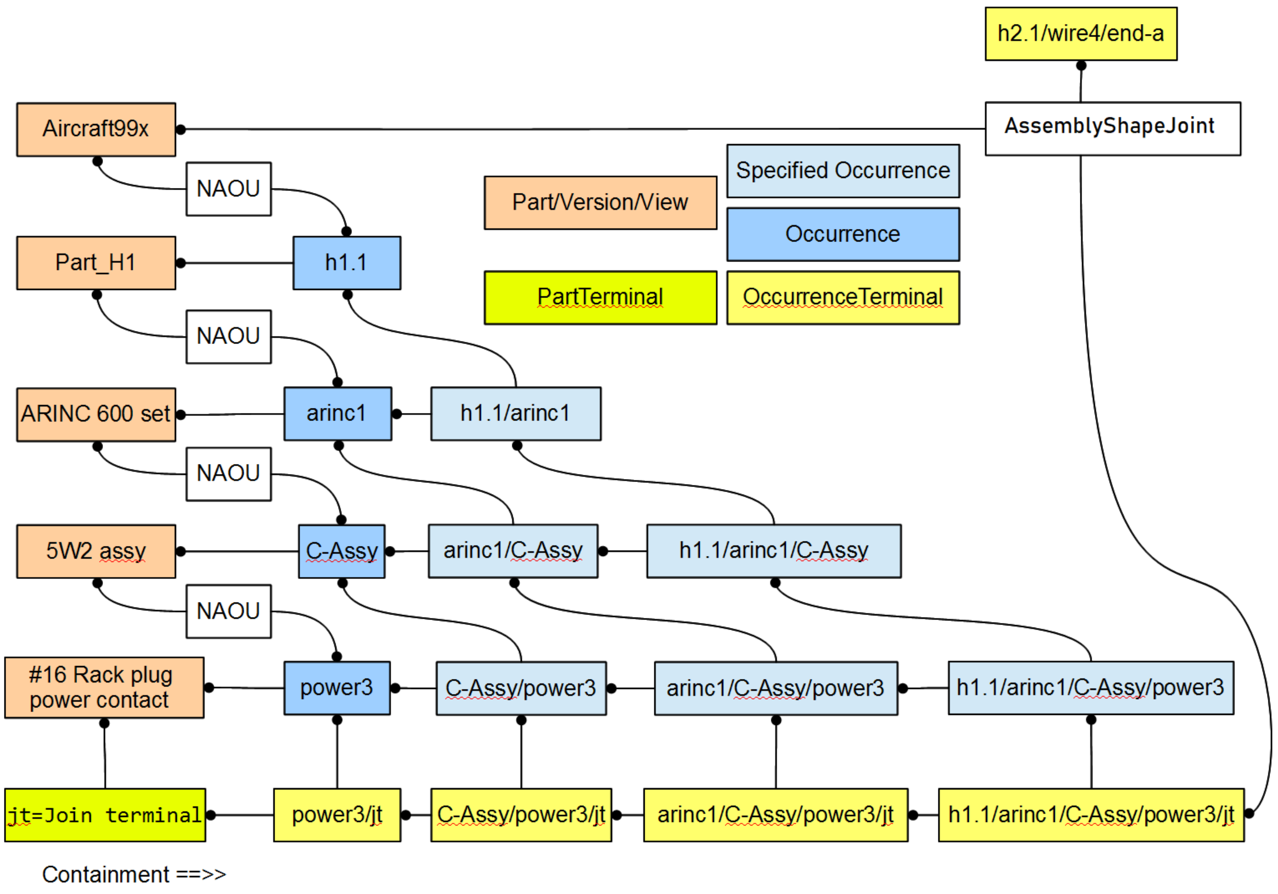


Figure 17: SpecifiedOccurrence Tree with OccurrenceTerminals

Now the terminal “jt” is made visible on the top assembly level and it can there be connected by an AssemblyShapeJoint to another terminal “h2.1/wire1/end-a” for which a similar structure will be build up.

Please note that the flattening of the hierarchical structure into a flat structure is essential because:

- the “5W2 assy” might contain another occurrence “power4” of an “#16 Rack plug power contact” (which is the case in our example)
- the “ARINC 600 set” might contain another occurrence “D-Assy” of an “5W2 assy”
- the “Part_H1” might contain another occurrence “arinc2” of an “ARINC 600 set”

- the “ Aircraft99x” might contain another example “h1.2” of an “Part_H1”

If all this would be the case, an AssemblyShapeJoint on the top assembly level would need to be specific as to which one of the following “jt” OccurrenceTerminals a connection is made:

- “h1.1/arinc1/C-Assy/power3”
- “h1.1/arinc1/C-Assy/power4”
- “h1.1/arinc1/D-Assy/power3”
- “h1.1/arinc1/D-Assy/power4”
- “h1.1/arinc2/C-Assy/power3”
- “h1.1/arinc2/C-Assy/power4”
- “h1.1/arinc2/D-Assy/power3”
- “h1.1/arinc2/D-Assy/power4”
- “h1.2/arinc1/C-Assy/power3”
- “h1.2/arinc1/C-Assy/power4”
- “h1.2/arinc1/D-Assy/power3”
- “h1.2/arinc1/D-Assy/power4”
- “h1.2/arinc2/C-Assy/power3”
- “h1.2/arinc2/C-Assy/power4”
- “h1.2/arinc2/D-Assy/power3”
- “h1.2/arinc2/D-Assy/power4”

10.6.1 Example: AssemblyShapeJoint in Hierarchical Assembly

xxx

10.7. Flat Assembly Structure with geometrically constrained Occurrences

xx TBD, see HarnessExample_flat.xml

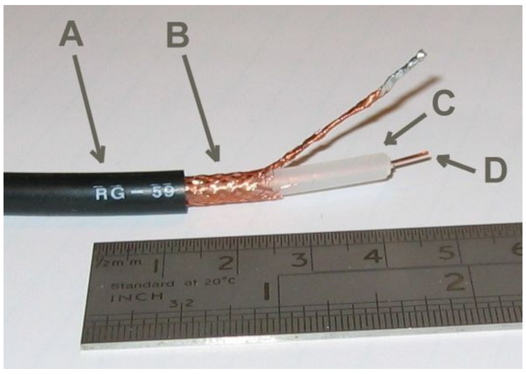
Some connectors have a quite complex assembly structure consisting of contacts, inserts, housing, back-shell etc. Instead of using hierarchical assemblies some system uses a flat assembly structure where these components all show up directly in the main WiringHarnessAssemblyDesign. Having several connectors there is a need to know which connector components belong together. This information can be provided by either a pure mechanical AssemblyShapeJoint or AssemblyShapeConstraint. The difference between these two concepts is that an AssemblyShapeJoint provides some information on how the joined components hold together (e.g. by soldering) while an AssemblyShapeConstraint only provides the information that some objects should geometrically touch in some way, but without saying how this is achieved.

10.8. Multi terminal connections

A coax cable and connectors typically often of a single shield and single wire/contact. For practical reason only a single connection is used for data exchange, and sender and receiver will understand that it is in fact 2 connections. Other examples are e.g.

- power connection, consisting of phase, neutral and GND
- USB-C connection with 24 contacts in 12 pairs
- ethernet connections consisting of 4 twisted cable pairs, so 8 connections

Here a detailed example for a coax cable of type RG-58 (50 Ohms) and a corresponding male and female connector



xxx

cable of `type`

11.Cable and Wire Occurrences

Let's start with the IEC definitions of what a wire and cable is:

- **wire** [IEV ref 151-12-28]: flexible cylindrical conductor, with or without an insulating covering, the length of which is large with respect to its cross-sectional dimensions
  Note – The cross-section of a wire may have any shape, but the term "wire" is not generally used for ribbons or tapes.
- **cable** [IEV ref 151-12-38]: assembly of one or more conductors and/or optical fibres, with a protective covering and possibly filling, insulating and protective material

For the purpose of AP242 the definitions may be used in a slightly different way. Important is that a wire has exactly one conductor, but it might be rigid (not flexible). And it is recommended to use cable only if there are two or more conductors.

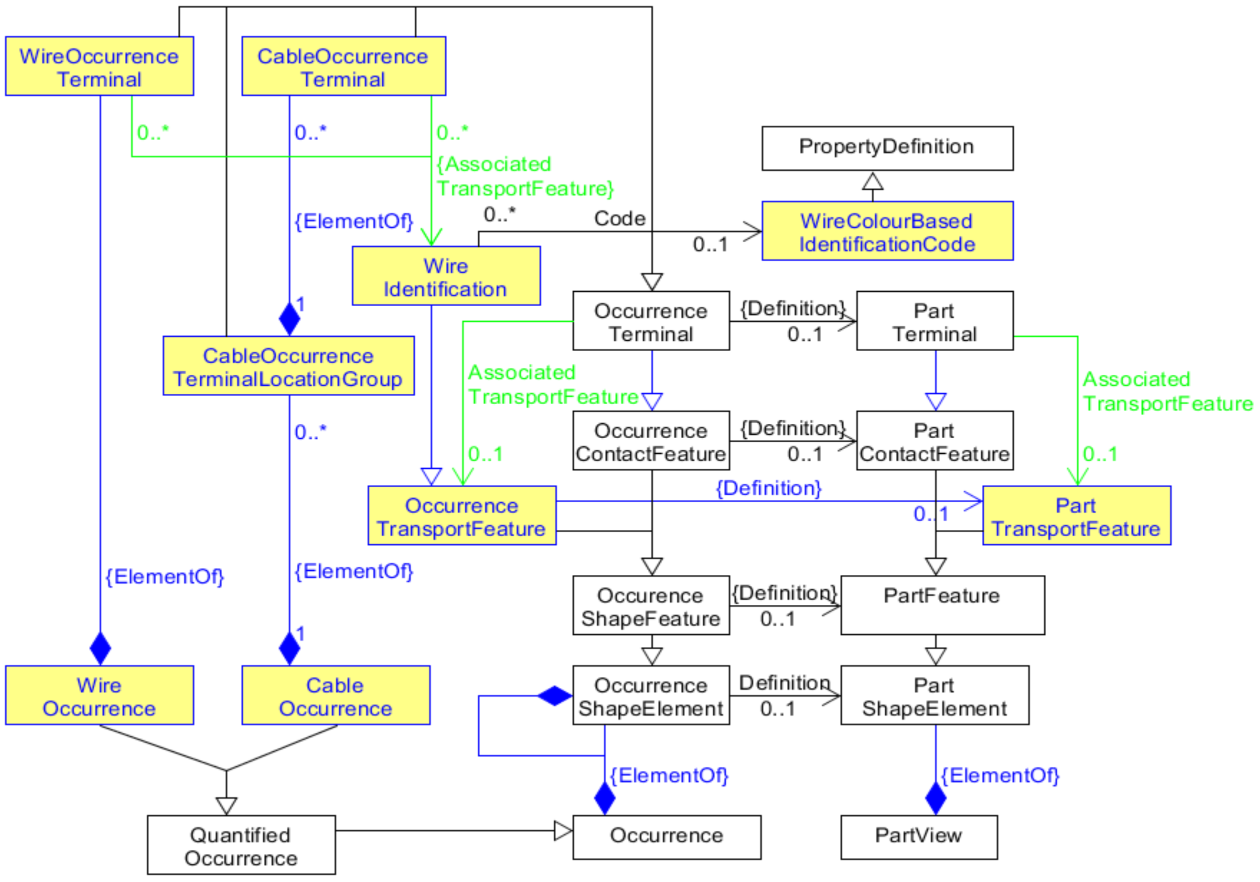


Figure 18: Terminals & TransportFeatures for Wires & Cables

Cables and Wires as raw materials are treated as Parts with the PartTypes attribute set to the PartCategoryEnum values “cable” respectively “wire”. Cables and wires are typically produced and sold in big lengths that are coiled. The length is often much bigger that the length needed for a particular usage and so a piece of wire or cable has to be cut from the coil in a certain length. Because of this the Part for the raw material get for PartType the additional PartCategoryEnum values “raw material by length”.

If the exact cut length is know it can directly be specified as a quantity with unit, e.g. 1.5 m. Otherwise a criterion can be specified e.g. “not determined in engineering; to be defined by manufacturer”.

CableOccurrence and WireOccurrence are Occurrences of Parts with a PartTypes that has the *PartCategoryEnum* values “cable” respectively “wire”. Because each *CableOccurrence* and *WireOccurrence* has a particular length they are subtypes of *QuantifiedOccurrence*.

So far we looked only to *OccurrenceTerminals* that are defined by *PartTerminals*. There is another class of *OccurrenceTerminals* that are defined locally as part of an OccurrenceTransportFeature which is defined by a PartTransportFeature.

Because the raw cable and wire material are typically rolled up and protected by some isolation, it is not possible to identify any particular *PartTerminal* on that raw material. So instead of identifying *PartTerminals* we only identify conductors (one or several) using *WireIdentification*. Both *PartTerminal* and *WireIdentification* are subtypes of *PartShapeElement*. A *WireIdentification* has a pre-defined property named *Code* for the “wire colour-based identification code”.

Cable and wire parts are raw material; they do no have specific *PartTerminals*. *CableOccurrences* and *WireOccurrences* on the other hand have terminals. These terminals are derived from the *WireIdentifications* of the raw materials.

In the same way we could have e.g. a GND-bar and define any number of terminals on it.

12. Cross Section of Cables or Harness Segments

When cutting through a harness segment perpendicular to its centre line we will see cables, wires, and other auxiliary materials such as isolations, shields, protections and more. The following data model is provided to describe such a structure.

The data model defines a directed acyclic graph, with CrossSectionalOccurrenceShapeElements as the leave elements that are defined by CrossSectionalPartShapeElements. The higher level nodes and root nodes in this graph are:

- CrossSectionalAlternativePartShapeElement
- TwistedCrossSectionalGroupShapeElement
- CrossSectionalGroupShapeElement
- TwistedCrossSectionalGroupShapeElement
- CrossSectionalGroupShapeElementWithTubularCover
- CrossSectionalGroupShapeElementWithLacing

The cross-section of a HarnessSegment can be defined by any ones of these nodes. However the definition of any of these nodes and leaves is not restricted to a single HarnessSegment. These nodes may span several segments and are therefore defined and reusable for the whole harness.

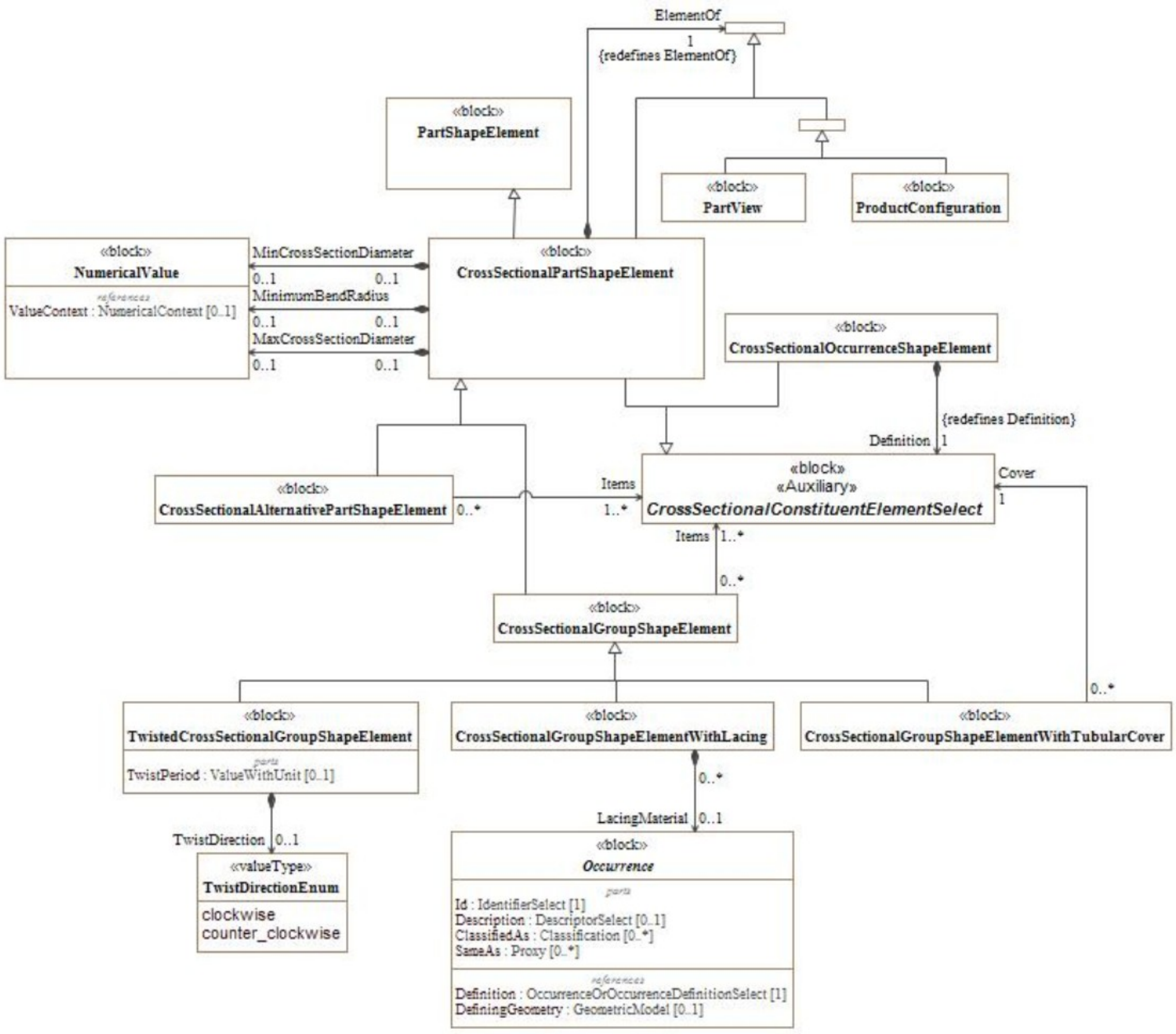

ElementOf
1
{redefines ElementOf}
«block» PartShapeElement
«block» PartView
«block» ProductConfiguration
«block» NumericalValue
references
ValueContext : NumericalContext [0..1]
MinCrossSectionDiameter
0..1 0..1
MinimumBendRadius
0..1 0..1
MaxCrossSectionDiameter
0..1 0..1
«block» CrossSectionalPartShapeElement
«block» CrossSectionalOccurrenceShapeElement
{redefines Definition}
Definition 1
«block» «Auxiliary» CrossSectionalConstituentElementSelect
Cover 1
Items 1..*
«block» CrossSectionalAlternativePartShapeElement 0..*
Items 1..*
0..*
«block» CrossSectionalGroupShapeElement
0..*
«block» TwistedCrossSectionalGroupShapeElement
parts
TwistPeriod : ValueWithUnit [0..1]
«block» CrossSectionalGroupShapeElementWithLacing
«block» CrossSectionalGroupShapeElementWithTubularCover
TwistDirection 0..1
«valueType» TwistDirectionEnum
clockwise
counter_clockwise
0..*
LacingMaterial 0..1
«block» Occurrence
parts
Id : IdentifierSelect [1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
SameAs : Proxy [0..*]
references
Definition : OccurrenceOrOccurrenceDefinitionSelect [1]
DefiningGeometry : GeometricModel [0..1]

Figure 19: Cross-sections of extruded structures

A CrossSectionalPartShapeElement, subtype of PartShapeElement represents an extruded shape along some centre-line of a part. The minimum bend radius for this shape (to the centre-line) as well as the minimal and maximal cross sectional diameter of the shape is provided. The exact geometry of the cross sectional ShapeElement (e.g. circular or rounded rectangle) is not specified as this may vary along the centre line. The boolean attribute Outer specifies if TRUE the outer shape along the centre line. If FALSE an inner shape is provided. A PartView can have at most one CrossSectionalPartShapeElement with value TRUE for Outer, but it may have zero, one or more CrossSectionalPartShapeElements with the value FALSE.

A CrossSectionalOccurrenceShapeElement is an OccurrenceShapeElement defined by a CrossSectionalPartShapeElement for an Occurrence of a Part. CrossSectionalOccurrenceShapeElements are the leave elements from which the cross section of a harness segment is build up.

A CrossSectionalConstituentElementSelect defines a select of either a CrossSectionalPartShapeElement or a CrossSectionalOccurrenceShapeElement.

A CrossSectionalAlternativePartShapeElement is a CrossSectionalPartShapeElement that defines that one of two or more CrossSectionalConstituentElementSelects is to be used as alternatives in a cross section.

A CrossSectionalGroupShapeElement is a CrossSectionalPartShapeElement that groups one or more items defined by CrossSectionalConstituentElementSelect. How the grouping is realised is not defined.

A TwistedCrossSectionalGroupShapeElement is a CrossSectionalGroupShapeElement where the items are grouped by twisting. The twist direction and period can be specified.

A CrossSectionalGroupShapeElementWithTubularCover is a CrossSectionalGroupShapeElement where the items are grouped by covering them all around with a specified side of material from a specific side of this material.

CrossSectionalGroupShapeElementWithLacing is a CrossSectionalGroupShapeElement where the items are laced together by some material. The details of lacing such as the period length are not specified.

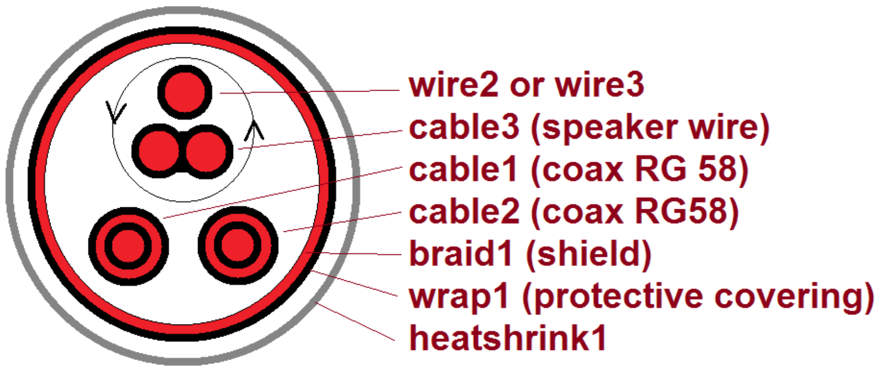


Figure 20: Example cross-section for harness segments S2-1 and S3-1

Explanation of the cross sectional structure in the provided example:

- for the wire, coax cable RG 48, and the Speaker Cable no detailed information is provided about the inner structure, insulation etc. So only the **outer** CrossSectionalPartShapeElement is provided for these parts, together with corresponding CrossSectionalOccurrenceShapeElements for the Occurrences. See the upper part of table 1
- for the Braid, Wrap and HeatShrink the **inner** CrossSectionalPartShapeElement and corresponding CrossSectionalOccurrenceShapeElement is needed as they need to go over something else. See the lower part of table 1.
- wire2 in segment S2 is equivalent to wire3 in segment S3; so these are taken as alternatives for a AlternativeLongitudinalExtendPart (_313001)

- the alternative of wire2 and wire3 are twisted with cable3, covered by TwistedCrossSectionalGroupShapeElement (_313002). The twist-direction is clockwise and the length of the period is 5 cm.
- the result of the upper twist is together with cable1 and cable2 grouped together and covered by braid1 for shielding purpose by CrossSectionalGroupShapeElementWithTubularCover (_313004).
- the group covered by braid1 is protected by wrap1 through CrossSectionalGroupShapeElementWithTubularCover (_313005).
- the group protected by wrap1 is further protected by heatshrink1 through CrossSectionalGroupShapeElementWithTubularCover _313006
- in some places (S2-2) the heatshrink1 is directly on braid1, CrossSectionalGroupShapeElementWithLacing (_313008).

In table4 and table5 the details of the cross sectional structure used in H1 are provided.

Name	**Occurrence**	**in/out**	**uid**
WIRE,ELEC,COMP,SNGL CONDUCTOR,150 DEG C	Part	outer	_101020
wire1	_201004	outer	_201009
wire2	_201104	outer	_201109
wire3	_201204	outer	_201209
RG 58	Part	outer	_102020
cable1	_202006	outer	_202007
cable2	_202106	outer	_202107
Speaker wire	Part	outer	_104020
cable3	_204006	outer	_204007
Braid 1/2inch	Part	inner	_120003
braid1	_220005	inner	_220006
Wrap	Part	inner	_121003
wrap1	_221005	inner	_221006
HeatShrink	Part	inner	_122003
heatshrink1	_222005	inner	_222006

Table 5: Cross-sectional ShapeElements of "bought" parts

Type	**uid / uidRef**	**Segment**	**comment**
AlternativeLongitudinalExtendPart	uid _313001	internal	wire2 or wire3
item	uidRef _201109		wire2/outer
item	uidRef _201209		wire3/outer
TwistedCSGSE	uid _313002	internal	twist (wire2 or wire3) with cable3
item	uidRef _313001		wire2 or wire3
item	uidRef _204007		cable3/outer
CSGSEWithTubularCover	uid _313004	S2-3,S3-2	braid1 on (twist...)
item	uidRef _313002		twist...
item	uidRef _202007		cable1/outer
item	uidRef _202107		cable2/outer
cover	uidRef _220006		braid1/inner
CSGSEWithTubularCover	uid _313005	internal	wrap1 on braid1 ...
item	uidRef _313004		braid1 on (twist...
cover	uidRef _221006		wrap1/inner
CSGSEWithTubularCover	uid _313006	S2-1,S3-1	heatshrink1 on wrap1
item	uidRef _313005		wrap1 on ...
cover	uidRef _222006		heatshrink1/inner
CSGSEWithTubularCover	uid _313007	S2-2	heatshrink1 on braid1 ...
item	uidRef _313004		braid1 on (twist...
cover	uidRef _222006		heatshrink1/inner
CSGSEWithLacing	uid _313008	S5	Lacing of cable1, cable2 & wire3
item	uidRef _201209		wire3/outer
item	uidRef _202007		cable1/outer
item	uidRef _202107		cable2/outer

Table 6: Cross-sectional structure definitions in H1

13. HarnessNode and HarnessSegment

A *WiringHarnessAssemblyDesign* is an *AssemblyDefinition* in which all the physical elements and the physical connectivity of an electrical harness is defined. Unless the harness has a very primitive structure, it refers to a *EdgeBasedTopologicalRepresentationWithLengthConstraint* that defines the topological structure.

A *HarnessSegment* is a kind of *PartShapeElement* for a *WiringHarnessAssemblyDesign* that links an *Edge* from the topological representation of the harness to a *CrossSectionalGroupShapeElement*. For manufacturing purpose the length of a referenced *Edge* can be overridden by the ForcedLength attribute.

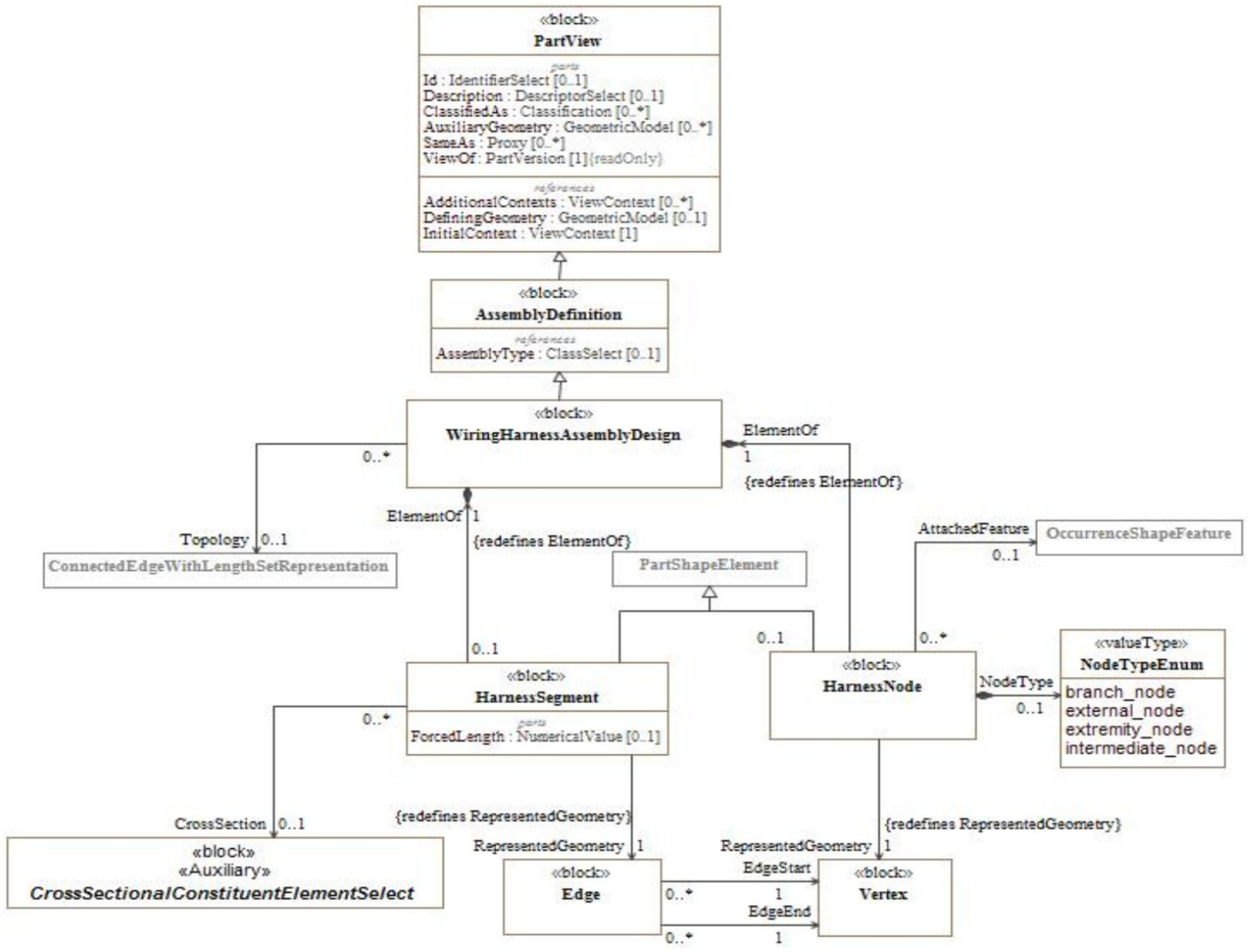

«block»
PartView
parts
Id : IdentifierSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
AuxiliaryGeometry : GeometricModel [0..*]
SameAs : Proxy [0..*]
ViewOf : PartVersion [1]{readOnly}
references
AdditionalContexts : ViewContext [0..*]
DefiningGeometry : GeometricModel [0..1]
InitialContext : ViewContext [1]
«block»
AssemblyDefinition
references
AssemblyType : ClassSelect [0..1]
«block»
WiringHarnessAssemblyDesign
ElementOf
1
{redefines ElementOf}
0..*
Topology
0..1
ConnectedEdgeWithLengthSetRepresentation
ElementOf
1
{redefines ElementOf}
PartShapeElement
AttachedFeature
0..1
OccurrenceShapeFeature
0..1
0..1
0..*
«block»
HarnessSegment
parts
ForcedLength : NumericalValue [0..1]
«block»
HarnessNode
NodeType
0..1
«valueType»
NodeTypeEnum
branch_node
external_node
extremity_node
intermediate_node
0..*
CrossSection
0..1
«block»
«Auxiliary»
CrossSectionalConstituentElementSelect
{redefines RepresentedGeometry}
RepresentedGeometry
1
«block»
Edge
RepresentedGeometry
1
{redefines RepresentedGeometry}
EdgeStart
1
0..*
EdgeEnd
1
0..*
«block»
Vertex

Figure 21: WiringHarnessAssemblyDesign with HarnessSegment and -Node

A *HarnessNode* is another kind of *PartShapeElement* for a *WiringHarnessAssemblyDesign* that links a *Vertex* from the topological representation to an *OccurrenceShapeElement* that represents an aspect of an *Occurrence*. With the attribute NodeType it is possible to characterise the kind of node:

- branch node: A node that is used by more than one Edge or HarnessSegment. The nodes N3 and N4 of the example are of this kind.
- extremity node: A node that is used by a single Edge or HarnessSegment. The nodes N1, N2, N5 and N6 of the example are of this kind.
- external node: A node in a partial harness design that is used by a single Edge or HarnessSegment and that is indicating that the harness continues beyond that node in another design. The node X4 in the example is of this kind for the sub-harnesses H1.a and H1.b.
- intermediate node: Additional nodes defined on Edges. The nodes X1, X2, X3, X4 (for H1), X5 and X6 of the example are of this type.

The main topological structure of a wiring harness is defined by the edges and vertices of the EdgeBasedTopologicalRepresentationWithLengthConstraint. The additional HarnessSegment and Harness Node objects have to be used when there:

- is a need for additional properties;
- to link with cross-sectional structure by HarnessSegment.CrossSection;
- to link with a feature of a connector, fastener or other rigid component by HarnessNode.AttachedFeature.

14. Representation & Geometry Models

Figure 22 shows the data model for *Representation* with subtypes *GeometricRepresentation* and -*Model* and relationships between them, including transformations and associations. A Representation is a collection of RepresentationItems for which many subtypes are defined (e.g. *AxisPlacement*, *Curve*, *CartesianPoint*, *Edge*, *Vertex*, *Path* ...). *RepresentationItems* are defined within a *RepresentationContext* and can therefore be used by the Representations of the same context. A *RepresentationItem* is included in a *Representation* either directly (via the *Items* attribute or indirectly because it it referenced by another *RepresentationItem*. E.g. when a *Representation* contains an *Edge*, it also contains the two *Vertices* called out by its *EdgeStart* and *EdgeEnd* attribute. In this tutorial we only explain the representation specializations that are needed for electrical harness.

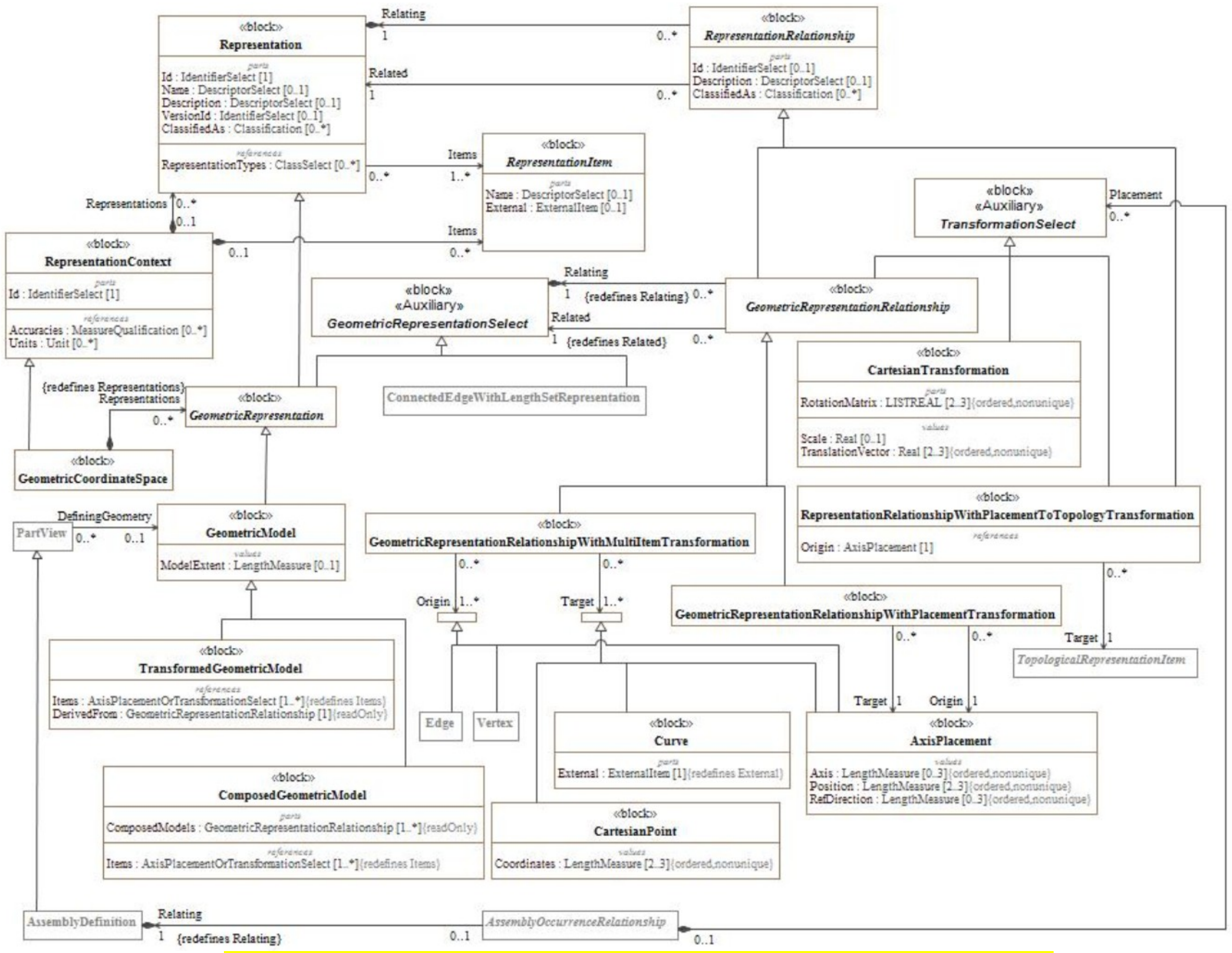

«block» Representation
Id : IdentifierSelect [1]
Name : DescriptorSelect [0..1]
Description : DescriptorSelect [0..1]
VersionId : IdentifierSelect [0..1]
ClassifiedAs : Classification [0..*]
RepresentationTypes : ClassSelect [0..*]
«block» RepresentationRelationship
Id : IdentifierSelect [0..1]
Description : DescriptorSelect [0..1]
ClassifiedAs : Classification [0..*]
Relating
Related
«block» RepresentationItem
Name : DescriptorSelect [0..1]
External : ExternalItem [0..1]
Items
Representations
«block» RepresentationContext
Id : IdentifierSelect [1]
Accuracies : MeasureQualification [0..*]
Units : Unit [0..*]
«block» «Auxiliary» TransformationSelect
Placement
«block» «Auxiliary» GeometricRepresentationSelect
«block» GeometricRepresentationRelationship
{redefines Relating}
{redefines Related}
ConnectedEdgeWithLengthSetRepresentation
«block» CartesianTransformation
RotationMatrix : LISTREAL [2..3]{ordered,nonunique}
Scale : Real [0..1]
TranslationVector : Real [2..3]{ordered,nonunique}
{redefines Representations} Representations
«block» GeometricRepresentation
«block» GeometricCoordinateSpace
«block» RepresentationRelationshipWithPlacementToTopologyTransformation
Origin : AxisPlacement [1]
PartView
DefiningGeometry
«block» GeometricModel
ModelExtent : LengthMeasure [0..1]
«block» GeometricRepresentationRelationshipWithMultiItemTransformation
Origin
Target
«block» GeometricRepresentationRelationshipWithPlacementTransformation
TopologicalRepresentationItem
«block» TransformedGeometricModel
Items : AxisPlacementOrTransformationSelect [1..*]{redefines Items}
DerivedFrom : GeometricRepresentationRelationship [1]{readOnly}
Edge
Vertex
«block» Curve
External : ExternalItem [1]{redefines External}
«block» AxisPlacement
Axis : LengthMeasure [0..3]{ordered,nonunique}
Position : LengthMeasure [2..3]{ordered,nonunique}
RefDirection : LengthMeasure [0..3]{ordered,nonunique}
«block» ComposedGeometricModel
ComposedModels : GeometricRepresentationRelationship [1..*]{readOnly}
Items : AxisPlacementOrTransformationSelect [1..*]{redefines Items}
«block» CartesianPoint
Coordinates : LengthMeasure [2..3]{ordered,nonunique}
AssemblyDefinition
AssemblyOccurrenceRelationship
{redefines Relating}

Figure 22: GeometricRepresentation, Relationship and Transformation

GeometricCoordinateSystem is a subtype of *RepresentationContext* that defines a geometric coordinate system in which CartesianPoints, Vectors and others can be defined. The attribute *DimensionCount* of *GeometricCoordinateSystem* defines the number of axis of the coordinate system which is typically 2 for 2D or 3 for 3D. Further down we see a special case where a 1D coordinate system is used. *GeometricRepresentation*, a subtype of Representation, requires the use of a *GeometricCoordinateSystem*. It is further subtyped to *GeometricModel* that might be used as the *DefiningGeometry* of a *PartView*. A *GeometricModel* might be an *ExternalGeometricModel* that is available as a separate file; e.g. a STEP-p21 file.

A pair of *Representations* from either the same or different contexts can be related to each other by *RepresentationRelationship*. If the *Definitional* attribute is set to TRUE, the *Related Representation* becomes a definitional part of the *Relating Representation*. Note that the *Relating* attribute is covered as containment for the XML exchange.

Geometrically defined transformations between two *GeometricRepresenations* are defined by subtypes of Geometric*RepresentationRelationship:*

- most widely used in STEP is *GeometricRepresentationRelationshipWithPlacementTransformation* where the transformation is indirectly defined by placing an *AxisPlacement* from the related *Representation* (attribute *Origin*) onto an *AxisPlacement* (attribute *Target*) of the relating *Representation*.
- when scaling or mirroring is needed a GeometricRepresentationRelationshipWithCartesianTransformation is to be used

Relations between *GeometricRepresentations* of the same *GeometricCoordinateSpace* typically don't allow any transformations and so the *GeometricRepresentationRelationshipWithSameCoordinateSpace* is to be used.

For a general introduction and usage guide of these concepts see the "Recommended Practices for AP242 Business Object Model XML Assembly Structure". That document provides several alternatives on how to do assemblies with transformation. For the purpose of EWH it is recommended to use the method as explained in clause "7.3.2.1 Implicit Transformation". When using the method as described in clause '7.3.1 Template "Simplified Positioning Representation"' it won't be possible to link a 2D or 3D geometric representation with the topological representation of an EWH.

14.1. Harness Topology Representation

The topological structure of an electrical wire harness, is primarily build up from the root topological elements *Vertex* and *Edge*. An *Edge* defines a directed link between two *Vertices*, a *Start* and an *End-Vertex*. For the purpose to represent the topology structure of an electrical wire harness, a *Vertex* is associated to a geometric *Point* and an *Edge* is associated to a *Curve*. Because of this the subtypes *VertexPoint* and *EdgeCurve* or *SubEdge* are to be used.

In most traditional STEP geometric models the subtype *CartesianPoint* for a *Point* and any of the subtypes of *Curve* such as a straight line are used, so that the points and curves are fully located and described within a geometric coordinate space, either 2D or 3D. However an electrical wire harness is typically flexible and has no fixed geometric appearance. Because of this a 2D or 3D geometric coordinate space is meaningless to describe the flexible nature of a harness. The above mentioned subtypes *CartesianPoint*, straight line and others can not be used as there are no fixed coordinates or directions.

Because of this only the supertype *Point* or the subtype *PointOnCurve* are to be used for the attribute *VertexGeometry* of *VertexPoints.* For *EdgeCurves* only the subtype *BoundedCurveWithLength* is to be used for attribute *BasicCurve*. A *BoundedCurveWithLength* is a curve with a fixed length that is parametrised from 0 to the length of the curve. When *PointOnCurve* is used, then the underlying curve must be a *BoundedCurveWithLength*. For the purpose to define proper rules to ensure these constraint the subtype *EdgeBoundedCurveWithLength* has been introduced. When a SubEdge is used, the ParentEdge must either be another SubEdge (a "bigger" one) or an *EdgeBoundedCurveWithLength.*.

When using geometric entities such as *Point* or *Curve* within a *Representation* the use of a *RepresentationContext* that is a *GeometricCoordinateSpace* is enforced (see also ISO 10303-42). A *GeometricCoordinateSpace* requires the specification of a *DimensionCount* that is typically 2D or 3D. However as explained above a 2D or 3D coordinate space is not suitable to describe the flexible nature of a harness. It is therefore recommended to use for the *DimensionCount* the value 1 to ensure that it is not meant to be 2D or 3D. However the value 1 does not means that a harness is defined in a one dimensional coordinate space (even if a single *BoundedCurveWithLength* might suggest there is one dimension, think about what happens if there are several independent *BoundedCurveWithLength*).

The *EdgeBoundedCurveWithLengths* are grouped within a ConnectedEdgeSet that ensures that all edges are somehow connected by sharing a common VertexPoint. Such a ConnectedEdgeSet is the main

RepresentationItem of an *EdgeBasedTopologicalRepresentationWithLengthConstraint*, the specialized Representation for the topological structure of WiringHarnessAssemblyDesign.
An *EdgeBasedTopologicalRepresentationWithLengthConstraint* must contain exactly one ConnectedEdgeSet. Other Items of an *EdgeBasedTopologicalRepresentationWithLengthConstraint* can be Path and VertexPoint. They are constraint in the following way:

- the attribute VertexGeometry of the additional VertexPoints must refer to a PointOnCurve that has a its BasicCurves one of the *BoundedCurveWithLength of the EdgeBoundedCurveWithLength used in the* ConnectedEdgeSet or by one of the Paths (see below)
- the additional Paths must either be formed from the *EdgeBoundedCurveWithLength* within the ConnectedEdgeSet or are of type SubEdge

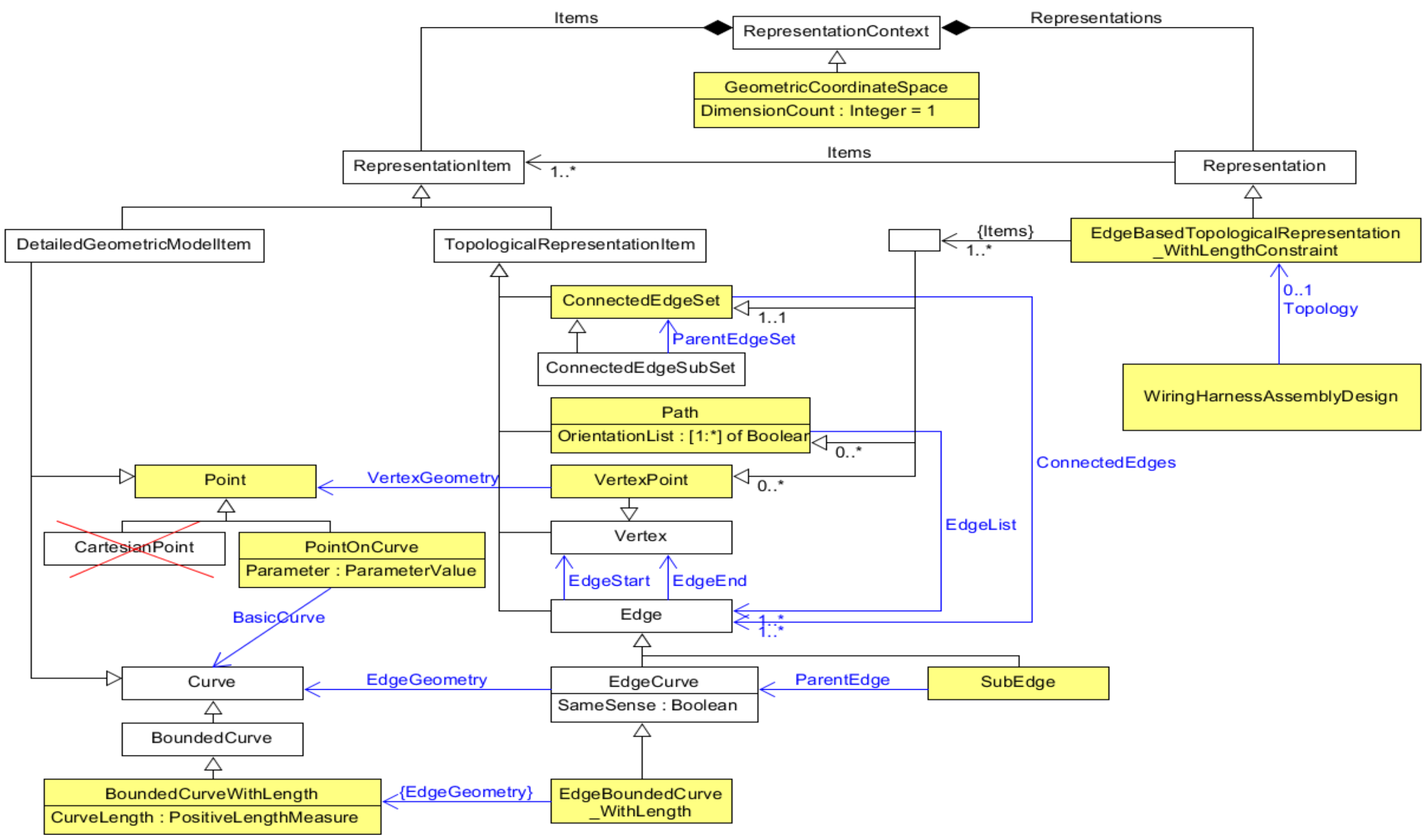


Figure 23: EdgeBasedTopologicalRepresentationWithLengthConstraint

The main topological structure consists of the following subtypes of *RepresentationItems*:

- *Vertex*
- *Edge*
- *EdgeWithLength*
- *ConnectedEdgeSet*
- *Path*

These items are grouped together in an *EdgeBasedTopologicalRepresentationWithLengthConstraint.*

An *Edge* is starting in one *Vertex* and is ending in another *Vertex*. Because of this an *Edge* has a direction. An *EdgeWithLength* is an *Edge* for which the length is defined. *Edges* that are connected with each other through common vertices can be grouped together in a *ConnectedEdgeSet*. A *ConnectedEdgeSet* defines a directed graph that may be cyclic (in other words, it may contain loops). Within this graph *Path*s can be specified that consists of an ordered list of connected *Edges* and a corresponding ordered list of boolean values for the orientation of the *Edges*. The *Edges* in a *Path* are constrained, so that the start *Vertex* of a following *Edge* must start at the end *Vertex* of a previous *Edge*. As needed the direction of an Edge can be inverted in its usage by the corresponding boolean attribute in the *OrientationList*.

Sub-topological structures can be defined on the main topological structure or a higher sub-topological structure with the following *RepresentationItems*:

- *SubEdge*
- *VertexOnEdge*
- *ConnectedEdgeSubSet*

A *SubEdge* is an *Edge* that represents a portion of a *ParentEdge.* A *SubEdge* might have the same start or same end Vertex or none of the Vertices of the parent *Edge*. In all other cases the start and end Vertex of a SubEdge must be VertexOnEdge.
A VertexOnEdge is a Vertex on a parent Edge that is an EdgeWithLength in a defined distance from either the start or the end Vertex. A VertexOnEdge must only be referenced by SubEdges with the same parent Edge.
A ConnectedEdgeSubSet is a ConnectedEdgeSet that defines a sub-set of connected Edges of a ParentEdgeSet.

A *EdgeBasedTopologicalRepresentationWithLengthConstraint* is a *Representation* that consists of exactly one *ConnectedEdgeSet* (with all the underlying edges and vertices) and any number of *Paths* on that *ConnectedEdgeSet* and any number of additional *SubEdge*s and V*ertexOfEdge*s defined on the *Edges* of the *ConnectedEdgeSet*.

As with all kinds of *Representation*s, the *RepresentationItem*s are defined under the *Items* section of a *RepresentationContext* and referenced in the Items section of a *EdgeBasedTopologicalRepresentationWithLengthConstraint*. Several *EdgeBasedTopologicalRepresentationWithLengthConstraint* can be defined under the same RepresentationContext, and thus allowing to share the same RepresentationItems.

14.2. Example Topological Harness Representation

Extract taken from HarnessExample_Hierarchical.xml:
The RepresentationContext _321000 is of type EdgeBasedTopologicalRepresentationWithLengthConstraints with a DimensionCount of 1. All length values are given in Metres. Within this context three EdgeBasedTopologicalRepresentationWithLengthConstraints are defined. The first one (_321010) defines the complete harness topology for the H1 example, while the other two defines sub-sets of the main one (_323010 for “H1-a” and _323010 for “H1-b”); see figure 1.

```
<RepresentationContext xsi:type="n0:GeometricCoordinateSpace" uid="_321000">
  <Id id="H1.x Harness topology context"/>
  <Units>
    <Unit uidRef="_100301"/>
  </Units>
  <Representations>
    <Representation
xsi:type="n0:EdgeBasedTopologicalRepresentationWithLengthConstraint" uid="_321010">
      <Id id="Topological representation of H1 harness"/>
      <Items>
        ... References to RepresentationItems
      </Items>
      <Representation
xsi:type="n0:EdgeBasedTopologicalRepresentationWithLengthConstraint" uid="_323010">
        <Id id="SubRep H1.b"/>
        <Items>
        ... References to RepresentationItems
        </Items>
      </Representation>
    </Representations>
    <Items>
    ... Definitions of RepresentationItems
    </Items>
  <DimensionCount>1</DimensionCount>
</RepresentationContext>

<Unit uid="_100301">
```

```
  <Name><ClassString>metre</ClassString></Name>
  <Quantity><ClassString>length</ClassString></Quantity>
</Unit>
```

Example of some topological representation items that are contained within *RepresentationContext/Items:*
The *ConnectedEdgeSet* _321020 consists of five *Edges* that are referenced. One of them is "_321023" that is has the specific subtype *EdgeBoundedCurveWithLength* and that is given the name S3. The underlying geometry of this edge is defined by the curve "_341023" with the specific subtype *BoundedCurveWithLength* with a length of 2.0 m. The *EdgeBoundedCurveWithLength* is an edge between two vertices. One of them (the *EdgeEnd)* is "_321043" that is of the specific subtype VertexPoint and that references for its geometry the Point "_341043".

The Path "_321065" consists of the two edges ("_321022" and "_321023"). The OrientationList contains the values TRUE and FALSE. This means that the first edge is followed as defined (from EdgeStart to EdgeEnd), while the second edge is followed in inverse order (EdgeEnd to EdgeStart). Consequently the path starts with the EdgeStart of the first edge and ends with the EdgeStart of the second edge.

```
<RepresentationItem xsi:type="n0:ConnectedEdgeSet" uid="_321020">
  <ConnectedEdges>
    <Edge uidRef="_321021"/>
    <Edge uidRef="_321022"/>
    <Edge uidRef="_321023"/>
    <Edge uidRef="_321024"/>
    <Edge uidRef="_321025"/>
  </ConnectedEdges>
</RepresentationItem>
<RepresentationItem xsi:type="n0:EdgeBoundedCurveWithLength " uid="_321023">
  <Name> <CharacterString>S3</CharacterString> </Name>
  <EdgeEnd uidRef="_321043"/>
  <EdgeStart uidRef="_321044"/>
  <EdgeGeometry uidRef="_341023"/>
  <SameSense>true</SameSense>
</RepresentationItem>
<RepresentationItem xsi:type="n0:BoundedCurveWithLength " uid="_341023">
  <CurveLength>2.0</CurveLength>
</RepresentationItem>
...
<RepresentationItem xsi:type="n0:VertexPoint" uid="_321043">
  <Name> <CharacterString>N3</CharacterString> </Name>
  <VertexGeometry uidRef="_341043"/>
</RepresentationItem>
<RepresentationItem xsi:type="n0:Point" uid="_341043"/>
...
<RepresentationItem xsi:type="n0:Path" uid="_321065">
  <Name> <CharacterString>S2-S3</CharacterString> </Name>
  <EdgeList>
    <Edge uidRef="_321022"/>
    <Edge uidRef="_321023"/>
  </EdgeList>
  <OrientationList>
    <Boolean>true</Boolean>
    <Boolean>false</Boolean>
  </OrientationList>
</RepresentationItem>
```

Example of sub-topological elements for the main topological elements defined above:
The VertexOnEdge _321051 with the name X1 is defined on the Edge _321023 (S3) at a distance of 0.8 m from Vertex _321044 (N4). The SubEdge _321035 (S3-2) is defined on _321023 (S3) between the start vertex _321051 (X1) and the end vertex _321044 (N4). Another SubEdge _321034 (S3-1) is defined on the same parent edge between the start vertex _321051 (X1) and the end vertex _321043 (N3). Because the main

edge S3 is 2.0 m long, a receiving system can deduce that the length of the S3-1 SubEdge is 1.2 m. The ConnectedEdgeSubSet defines a sub-set of Edges from the parent _321020 and contains the main edge _321022 (S2), the sub-edge _321034 (S-1) and others.

```
<RepresentationItem xsi:type="n0:ConnectedEdgeSubSet" uid="_322020">
  <ConnectedEdges>
    <Edge uidRef="_321021"/> <!-- S1 -->
    <Edge uidRef="_321022"/> <!-- S2 -->
    <Edge uidRef="_321034"/> <!-- S3-a -->
  </ConnectedEdges>
  <ParentEdgeSet uidRef="_321020"/>
</RepresentationItem>

<RepresentationItem xsi:type="n0:SubEdge" uid="_321034">
  <Name> <CharacterString>S3-1</CharacterString> </Name>
  <EdgeEnd uidRef="_321043"/>
  <EdgeStart uidRef="_321051"/>
  <ParentEdge uidRef="_c"/>
</RepresentationItem>

<RepresentationItem xsi:type="n0:VertexPoint" uid="_321051">
  <Name> <CharacterString>X1</CharacterString> </Name>
  <VertexGeometry uidRef="_341051"/>
</RepresentationItem>
<RepresentationItem xsi:type="n0:PointOnCurve" uid="_341051">
  <BasicCurve uidRef="_341023"/>
  <Parameter>0.8</Parameter>
</RepresentationItem>
```

14.3. Direction control of Edges and Paths

Both, Edges and Paths have a dedicated direction. For an Edge the direction is defined from the EdgeStart to the EdgeEnd. For a Path the direction is defined from one of the Vertices of the first Edge to one of the Vertices of the last Edge:

- If for the first Edge the OrientationList is TRUE, then the Path starts with its EdgeStart, otherwise with its EdgeEnd.
- If for the last Edge the OrientationList is TRUE, then the Path ends with its EdgeEnd, otherwise with its EdgeStart.

Two consecutive Edges in a Path must share a common VertexPoint. On whether this the EdgeStart or the EdgeEnd of these Edges depends on the corresponding values in the OrientationList:

- If the preceding Edge has an OrientationList value of TRUE, its EdgeEnd is used; otherwise its EdgeStart.
- If the succeeding Edge has an OrientationList value of TRUE, its EdgeStart is used; otherwise its EdgeEnd.

The following figure shows an example of a *Path* consisting of 3 *Edges* of type *EdgeBoundesCurveWithLength*:

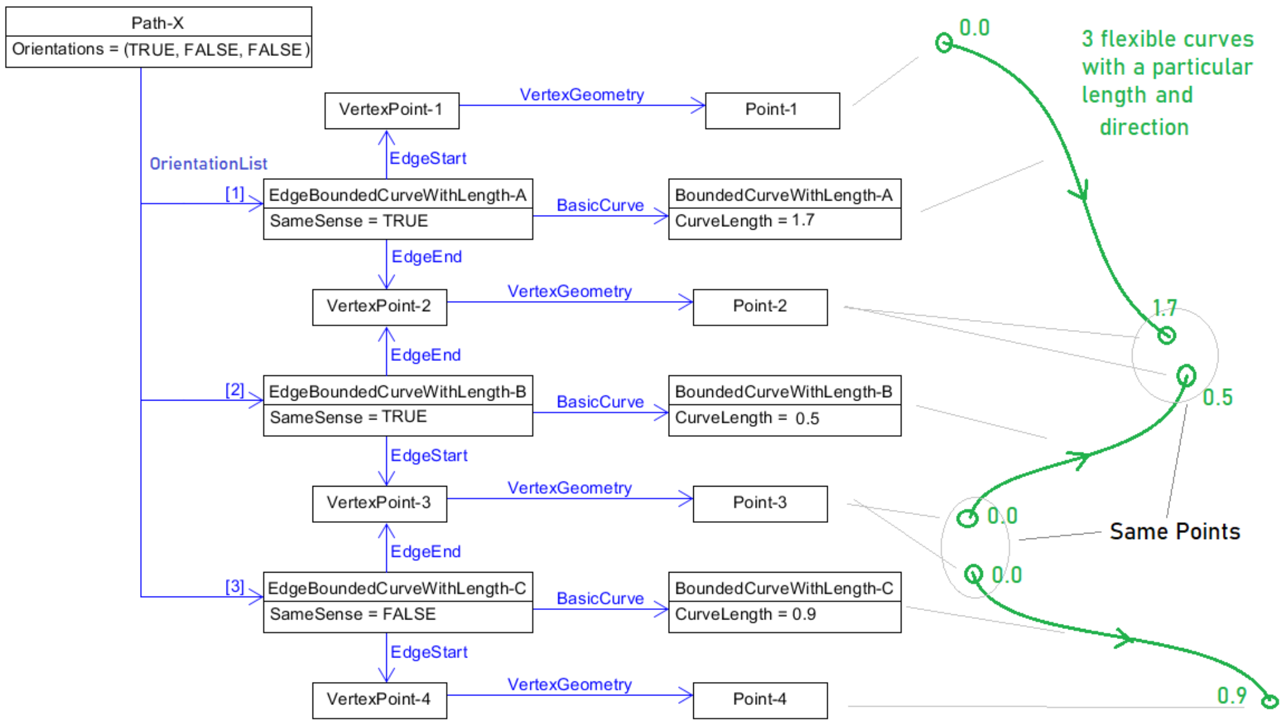


Figure 24: Example of Edge and Path directions

For the purpose of electrical wire harness the chosen directions of singleEdges and Paths don't matter. However the chosen directions of all the Edges and Paths must be consistent with each other.

14.4. External References for Geometry

As the AP242 DomainModel and its XML encoding do not support detailed geometric representations there is a need to reference traditional STEP files in p21 format that contain the needed geometry. The reference from the DomainModel XML file to the p21 file and then to specific entity instances in that files requires a sequence of steps.

1. a *FormatProperty* defines the *DataFormat* of the external file, e.g. one that is defined by "AP 10303-214".
2. a *DigitalFile* identifies the external file. If refers to the *FormatProperty* so that the type of external file is known on the XML level. With *FileLocations* the information where to find the file can be provided. Attribute *SourceId* of *FileLocationIdentification* is used for the file name.
3. *ExternalGeometricModel*, that is a wrapper on the XML level for an external geometry model, refers to the *DigtialFile* that contains the geometric model.
4. In the case that the external file contains more than one external geometric model, it is essential to specify which one. This is done through the attribute *ExternalItem*.
5. In the case that the external file is a STEP p21 file, *ExternalItem* shall be of type *ExternalEntityInstance* with the *Id* attribute specifying either the local instance ID that is used in the data section of the p21 file (e.g. "#1234") or the external anchor name that can be provided in the anchor section of a p21 file (available from edition 3 on).
6. an *ExternalGeometryModel* contain one or more *RepresentationItems*, some of which might be *ExternalRepresentationItems*. This can be use to replicate entity instances of an external p21 file on the XML level so that they can be references by other XML elements.
7. There are cases when it is not sufficient to just reference a particular entity instance in a p21 file but there is a need to follow a chain of entity instances that provide essential context information. This can be done by *NextInstanceForward* and *NextInstanceInverse*. Both specify the *AttributeName* to

follow and the NextInstance in the chain of instances. For *NextInstanceForward*, *AttributeName* specified an attribute of the current instance, while for *NextInstanceInverse*, *AttributeName* specifies an attribute of the *NextInstance*.

There had been a change from AP242 edition 1 to edition 2 on the attributes of *DigitalFile*. With edition 2 the attribute *FileLocation* was added (in green colour), replacing the attribute *Locations* (red colour) from edition 1. For an XML file generated according to AP242 edition 2, the *Locations* attribute must not be used.

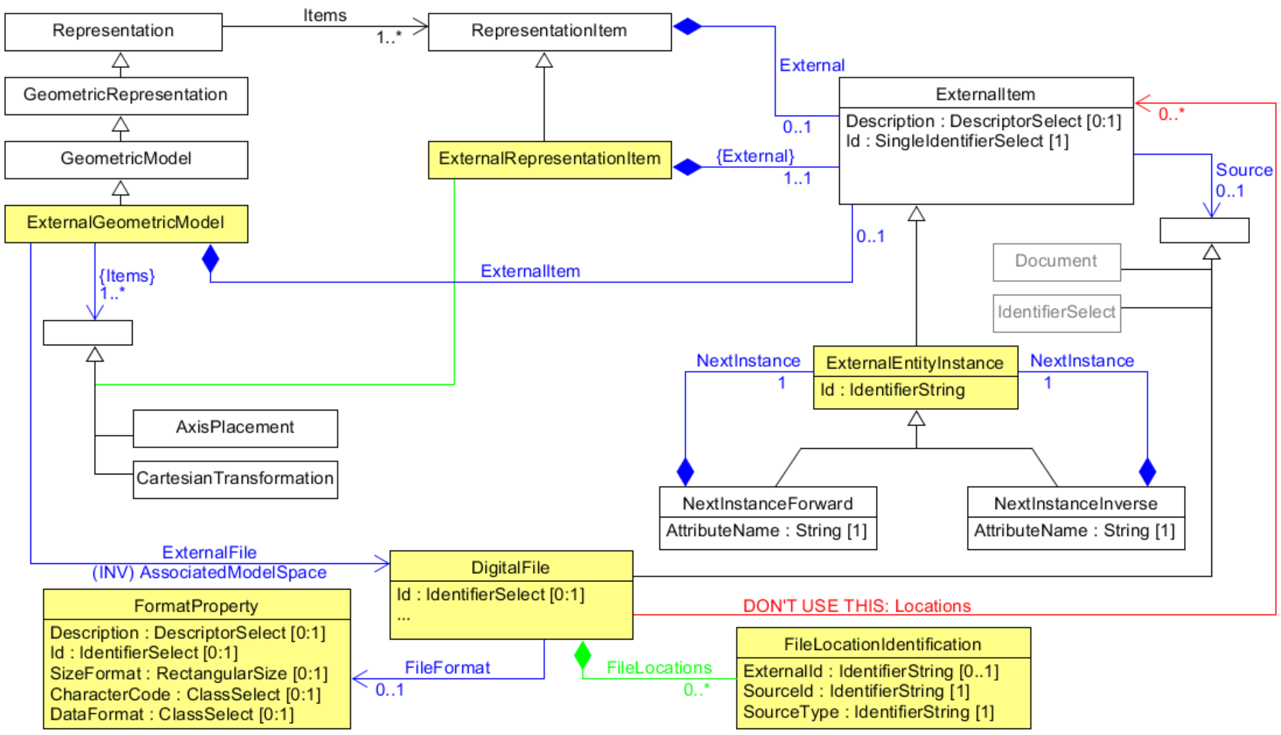

Representation
Items
1..*
RepresentationItem
External
ExternalItem
Description : DescriptorSelect [0:1]
Id : SingleIdentifierSelect [1]
0..*
GeometricRepresentation
GeometricModel
ExternalRepresentationItem
{External}
1..1
0..1
Source
0..1
ExternalGeometricModel
{Items}
1..*
ExternalItem
0..1
Document
IdentifierSelect
NextInstance
1
ExternalEntityInstance
Id : IdentifierString
NextInstance
1
AxisPlacement
CartesianTransformation
NextInstanceForward
AttributeName : String [1]
NextInstanceInverse
AttributeName : String [1]
ExternalFile
(INV) AssociatedModelSpace
DigitalFile
Id : IdentifierSelect [0:1]
...
FormatProperty
Description : DescriptorSelect [0:1]
Id : IdentifierSelect [0:1]
SizeFormat : RectangularSize [0:1]
CharacterCode : ClassSelect [0:1]
DataFormat : ClassSelect [0:1]
DON'T USE THIS: Locations
FileFormat
0..1
FileLocations
0..*
FileLocationIdentification
ExternalId : IdentifierString [0..1]
SourceId : IdentifierString [1]
SourceType : IdentifierString [1]

Figure 25: Referencing into External Models

14.4.1 External References to geometry files (as a whole)

For an explanation and example on how to use an *ExternalGeometricModel* to reference into a *DigitalFile* as a whole see the Recommended Practices for “Product & Assembly Structure”.

14.4.2 Example of External Element References into p21 file

For the purpose of electrical wire harness it is often essential to not only reference an external file that but also to reference to specific elements of an external file. This clause provides an example of how to reference an external STEP file and within this file reference to 3 specific entity instances, one that is a *representation* and two *representation_items*.

Extract taken from HarnessExample_Flat.xml:

1. *FileFormat* "_100300" defines a file type according to the “ISO 10303-214” standard.
2. *DigitalFile* "_103080" references this *FileFormat* and defines the *SourceId* “c-51864-1-af-3d.stp” as the name of the STEP file. It is assumed that this STEP file contains only a single *Representation*.
3. *GeometricCoordinateSpace* "_103091" contains a single *Representation* of type *ExternalAdvancedBrepShapeRepresentation* that specified as its external file the *DigitalFile* (see above). As the file may contain several representation instances, the exact one is specified by *ExternalItem*.
4. The *ExternalAdvancedBrepShapeRepresentation* contains 3 *RepresentationItems*. two of which are

of type *ExternalRepresentationItem* and references to some *ExternalEntityInstance* in the digital file. The third one is an *AxisPlacement* that is needed for positioning inside the XML file.

5. A PartView of some Part/PartVersion specifies the above *ExternalAdvancedBrepShapeRepresentation* as its RepresentedGeometry..
6. The *PartView* contains two *ShapeElements* (here of type *PartTerminals*) that defines the above *ExternalRepresentationItems* as their RepresentedGeometry respectively.

Through this chain of references the two ShapeElements are linked to two specific RepresentationItems in the external GeometryModel.

```
<FormatProperty uid="_100300">
  <DataFormat>
    <ClassString>ISO 10303-214</ClassString>
  </DataFormat>
</FormatProperty>
...
<!-- terminal lug geometric representation-->
<File xsi:type="n0:DigitalFile" uid="_103080">
  <FileFormat uidRef="_100300"/>
  <FileLocations>
    <FileLocationIdentification uid="_103081">
      <SourceId>c-51864-1-af-3d.stp</SourceId>
      <SourceType>file</SourceType>
    </FileLocationIdentification>
  </FileLocations>
</File>

<!--Geometry for terminal lug-->
<RepresentationContext xsi:type="n0:GeometricCoordinateSpace" uid="_103091">
  <Id id="/NULL"/>
  <Representations>
    <Representation
        xsi:type="n0:ExternalAdvancedBrepShapeRepresentation" uid="_103090">
      <Id id="c-51864-1-af-3d.stp"/>
      <Items>
        <RepresentationItem uidRef="_103092"/>
        <RepresentationItem uidRef="_103094"/>
      </Items>
      <ExternalFile uidRef="_103080"/>
      <ExternalItem xsi:type="n0:ExternalEntityInstance" uid="_103099">
        <Id id="#999"/>
      </ExternalItem>
    </Representation>
  </Representations>
  <Items>
    <RepresentationItem xsi:type="n0:ExternalRepresentationItem" uid="_103092">
      <External xsi:type="n0:ExternalEntityInstance" uid="_103093">
        <Id id="#521"/>
      </External>
    </RepresentationItem>
    <RepresentationItem xsi:type="n0:ExternalRepresentationItem" uid="_103094">
      <External xsi:type="n0:ExternalEntityInstance" uid="_103095">
        <Id id="#940"/>
      </External>
    </RepresentationItem>
    <RepresentationItem xsi:type="n0:AxisPlacement" uid="_103096">
      <Position>0.0 0.0 0.0</Position>
    </RepresentationItem>
  </Items>
  <DimensionCount>3</DimensionCount>
</RepresentationContext>
...
<Part uid="_103000">
  ...
  <Versions>
    <PartVersion uid="_103001">
      <Id id="Version 1"/>
      <Views>
        <PartView uid="_103002">
          ...
          <DefiningGeometry uidRef="_103090"/>
          ...
          <ShapeElement xsi:type="n0:PartTerminal" uid="_103003">
            ...
            <RepresentedGeometry uidRef="_103092"/>
            ...
          </ShapeElement>
          <ShapeElement xsi:type="n0:PartTerminal" uid="_103004">
            ...
            <RepresentedGeometry uidRef="_103094"/>
            ...
          </ShapeElement>
          ...
        </PartView>
      </Views>
    </PartVersion>
  </Versions>
</Part>
```

14.4.3 External Element Reference: chain of instances

TBD in a later edition

15. Transformations and Associations

There are different kinds of *RepresentationRelationship* for different purposes. In this clause we will compare the traditional ones for assembly structures of rigid component with the ones used for flexible structures such as electrical wire harness

15.1. Assembly structure of rigid components

In an assembly structure of rigid components a *NextAssemblyOccurrenceUsage* is used to relate a *SingleOccurrence* that is typically defined by a *PartView* of a piece part into another *PartView* that is an *AssemblyDefinition* of another Part that is the assembly.

In such a situation both *PartViews* may have the *DefiningGeometry* attributes set. The *PartView* of the piece part typically refers to an *ExternalGeometricModel* where all the details are defined. On the XML level only an *AxisPlacement* for the *Origin* of the model is provided (typically at coordinates 0/0/0 with no rotation). For the *PartView* of the assembly however there is typically no external geometric model available. Instead there is a *ComposedGeometricModel* that is the composition of all it's components with some transformation.

The geometric composition and transformation is defined by *GeometricRepresentationRelationshipWithPlacementTransformation* that relates the *ExternalGeometricModels* of the piece parts into the *ComposedGeometricModel* of the assembly. The transformation is defined by a pair of *AxisPlacements*, one that defines an *Origin* for the *ExternalGeometricModel* and one that defines the *Target* within the *ComposedGeometricModel* where a "copy" of the ExternalGeometricModel is virtually placed (this is done by a receiving system that visualize a ComposedGeometricModel).

As a piece part might show up several times in an assembly through several *SingleOccurrences*, it is essential to know which transformation applied for which SingleInstance. For this purpose the *Placement* attribute of a *NextAssemblyOccurrenceUsage* refers to the *GeometricRepresentationRelationshipWithPlacementTransformation* for a given *SingleOccurrence* within the assembly.

A *GeometricRepresentationRelationshipWithPlacementTransformation* that is used as described here must have the *Definitional* attribute set to TRUE as the *ExternalGeometricModel* becomes a definitional piece of the *ComposedGeometricModel*. There are other usages of *GeometricRepresentationRelationshipWithPlacementTransformation* where the Definitional attribute has to be set to FALSE; e.g. for a ToolPartRelationship between the PartViews of a screw and a screw-driver.

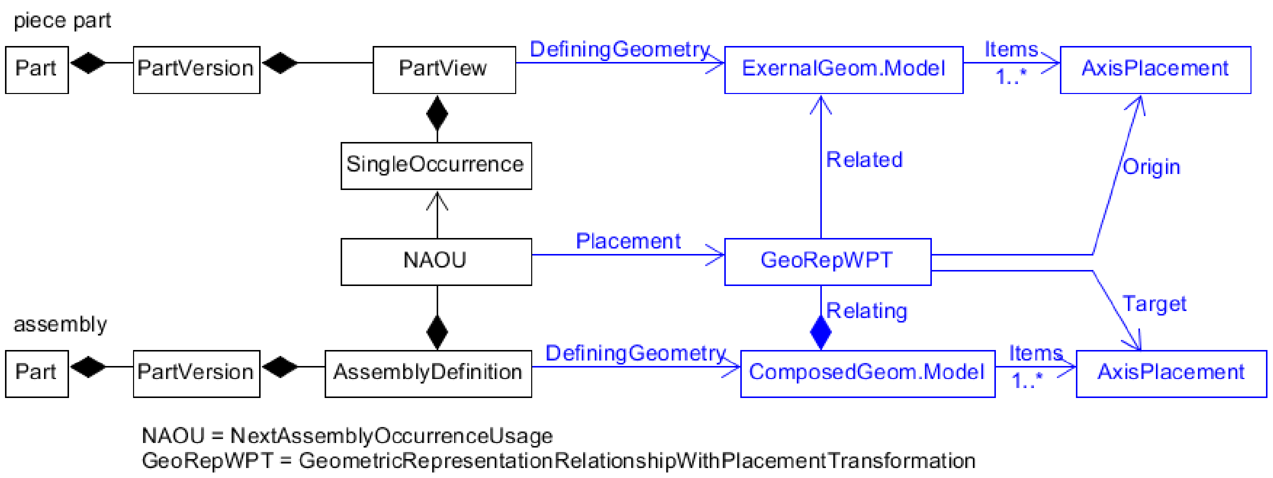


Figure 26: Simple assembly structure for a rigid component

15.2. Flexible topological and rigid 3D models of a Harness

The top object of an electrical wiring harness is a *Part* with the *PartCategoryEnum* "wiring harness".

The specific associations needed to represent the design of an electrical wiring harness are supported by *WiringHarnessAssemblyDesign* that is a specialization of *AssemblyDefinition* that is a *PartView*.

As explained in clause xxx the main geometric model of a WiringHarnessAssemblyDesign is defined by an EdgeBasedTopologicalRepresentationWithLengthConstraint that is referenced by the attribute Topology. In addition the WiringHarnessAssemblyDesign might have the attributes DefiningGeometry and AuxiliaryGeometry set to a 2D or 3D GeometricModels that represent the final intended shape of the harness when installed, or shape when the harness is delivered, or as manufactured on a foam board. However it must be clear that all these GeometricModels represent only one of of very many possible ones the wiring harness may have. Only when a wiring harness is installed in a higher assembly (e.g. an aircraft or vehicle) a fixed 3D model can be applied. But even for this case there might be variations, e.g. because part of a harness may end in a door that can move relative to other parts of a harness that are in the main part of the vehicle (see Kinematics capabilities of AP242). It is also to mention that for maintenance purpose the 3D shape of a harness will change, e.g. when opening a plug-socket connection.

For a full design documentation a *WiringHarnessAssemblyDesign* shall have an associated topological model in the form of a *EdgeBasedTopologicalRepresentationWithLengthConstraint* topology. The attribute *Topology* might not be set if the *EdgeBasedTopologicalRepresentationWithLengthConstraint* is not available or if the topology is too simple that it is not needed; e.g. in the case of a single cable wih two connectors at the ends.

Example taken from HarnessExample_Hierarchical.stpx:
The *WiringHarnessAssemblyDesign* "_311002" refers to the *EdgeBasedTopologicalRepresentationWithLengthConstraint* "_321010" by the attribute Topology and to the *ExternalAdvancedBrepShapeRepresentation* "_314090"by the attribute *DefiningGeometry*.

```
<Part uid="_311000">
  <Id id="Part_H1"/>
  <Name> <CharacterString>Electrical Harness example 1</CharacterString> </Name>
  <PartTypes>
    <PartCategoryEnum>wiring_harness</PartCategoryEnum>
    <PartCategoryEnum>discrete</PartCategoryEnum>
  </PartTypes>
  <Versions>
    <PartVersion uid="_311001">
      <Id></Id>
      <Views>
        <PartView xsi:type="n0:WiringHarnessAssemblyDesign" uid="_311002">"
uid="_311002">
          ...
          <DefiningGeometry uidRef="_314090"/>
          ...
          <Topology uidRef="_321010" />
        </PartView>
      </Views>
    </PartVersion>
  </Versions>
</Part>
```

15.3. Assembly structure of flexible components

For the purpose of electrical wiring harness with flexible components such as wires and cables, two additional *RepresentationRelationships* have been introduced, *GeometryToTopologyModelAssociation* and *TopologyToGeometryModelAssociation*. Both can be used as well by the *Placement* attribute of *NextAssemblyOccurrenceUsage*, but these new associations don't define a geometric transformation. Therefore the inherited *Definitional* attribute must be set to FALSE; the *Relating Representation* will not become part of the *Related* one.

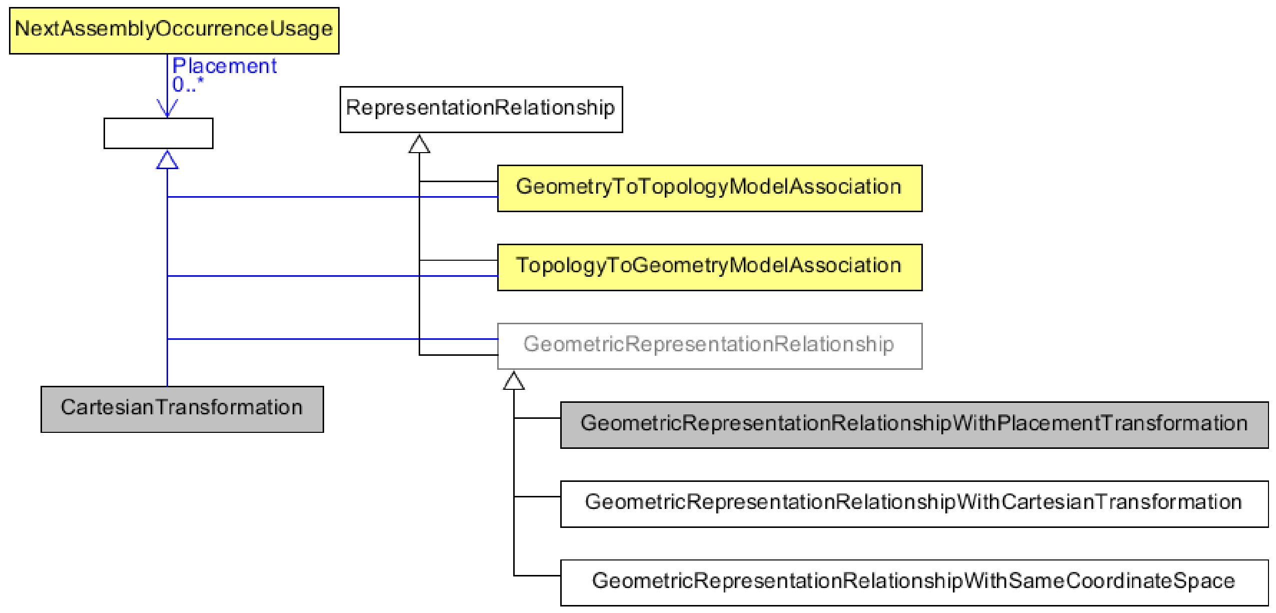

NextAssemblyOccurrenceUsage
Placement
0..*
RepresentationRelationship
GeometryToTopologyModelAssociation
TopologyToGeometryModelAssociation
GeometricRepresentationRelationship
CartesianTransformation
GeometricRepresentationRelationshipWithPlacementTransformation
GeometricRepresentationRelationshipWithCartesianTransformation
GeometricRepresentationRelationshipWithSameCoordinateSpace

Figure 27: RepresentationRelationships for Transformation and Association

Only specific kinds of RepresentationItems that are allowed as Origin and Target for *GeometryToTopologyModelAssociation* and *TopologyToGeometryModelAssociation*.

- For the *Origin* of a *GeometryToTopologyModelAssociation* or the *Target* of a *TopologyToGeometryModelAssociation* only the types *Edge* (subtype *EdgeBoundedCurveWithLength* or *SubEdge*), *Path* and *Vertex* (subtype *VertexPoint*) are allowd.
- For the *Target* of a *GeometryToTopologyModelAssociation* or the *Origina* of a *TopologyToGeometryModelAssociation* only the types *AxisPlacement, CartesianPoint, Curve, EdgeCurve or VertexPoint* are allowed.

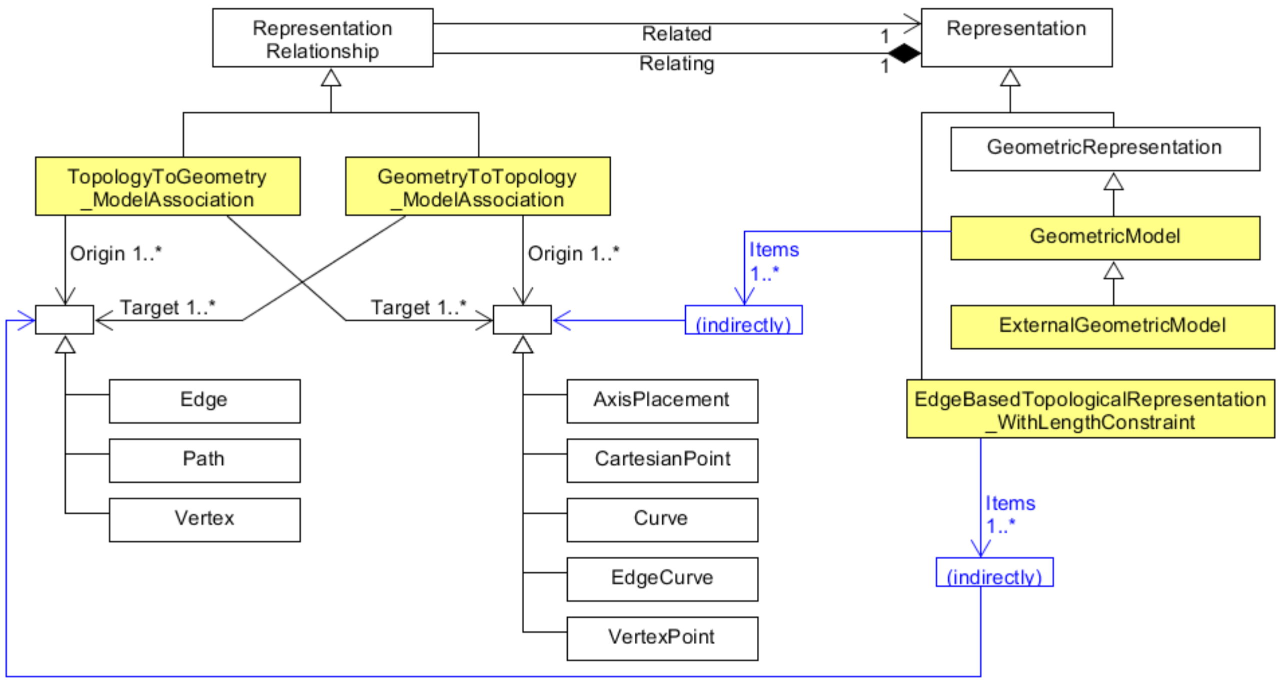


Figure 28: RepresentationItems used for Associations

15.4. Associating Models of rigid and flexible Part to the topological Model of a Wire Harness Model

This clause focuses on the application object **GeometryToTopologyModelAssociation** (G2TMAsso). It is used to associate the GeometricModels of occurrences of rigid and flexible parts to the topological items of a *EdgeBasedTopologicalRepresentationWithLengthConstraint* (EBTRepWLC) of a *WiringHarnessAssemblyDesign* (WHAD), see figure 30. The structure and usage of a G2TMAsso is very similar to a **GeometricRepresentationRelationshipWithPlacementTransformation** (GeoRepWPT) as they are both subtypes of the main supertype **RepresentationRelationship**. They consists of two pairs of associated objects. With the Related/Relating attributes the *Representations* are associated and with Origin/Target attributes the detailed *RepresentationItems* of these *Representations* are associated.

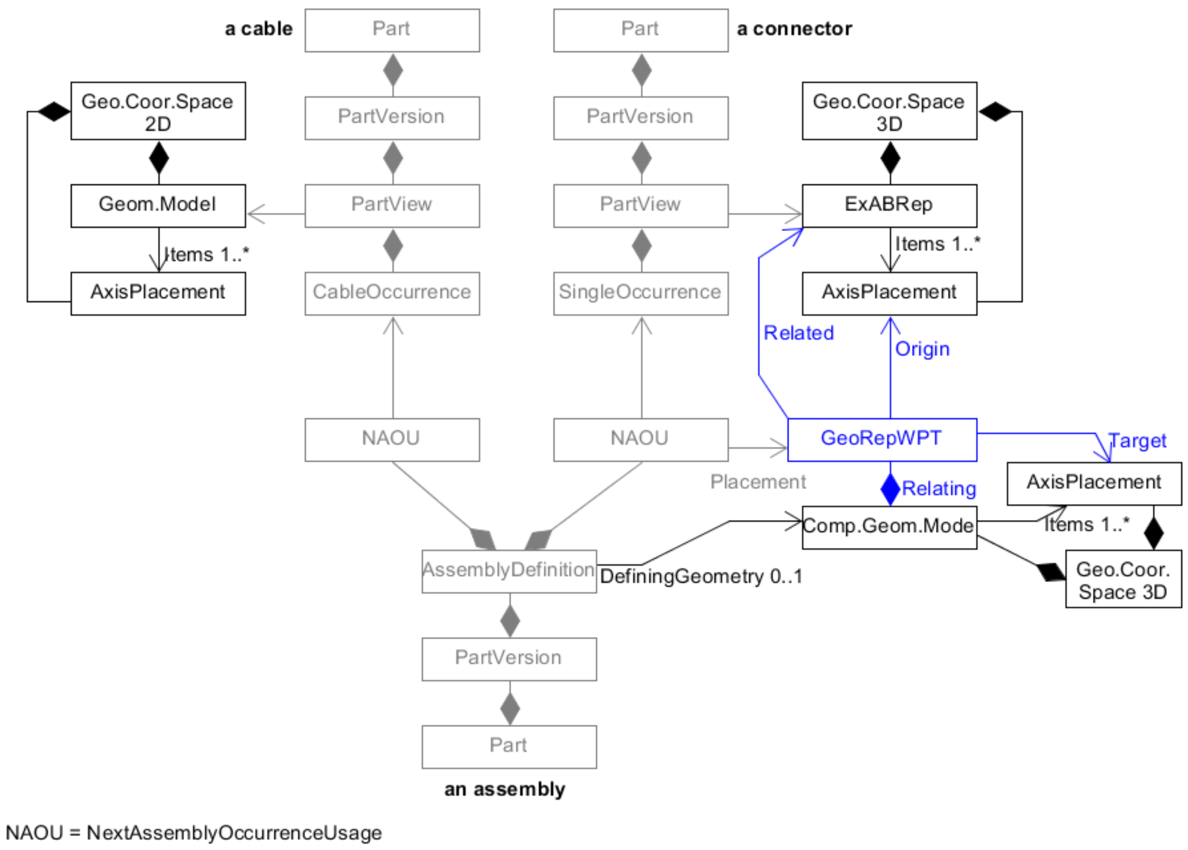


Figure 29: Transformation of a rigid part occurrences into an assembly

The default pattern of how *Occurrences* (Single or Quantified) are brought into an *AssemblyDesign* apply also for a WHAD. The *Single-* and *QuantifiedOccurrences* are brought into a WHAD through *NextAssemblyOccurrenceUsages* (NAOU). With it's attribute Placement, the NAOU is pointing to the corresponding G2TMAsso. This structure is important as there may be several occurrences of the same PartView and so it is possible to distinguish to association between these different occurrences.

The underlying *PartViews* of the Occurrences must have an associated *GeometryModel*. For a rigid Part/PartVersion/PartView such as a connector, the GeometricModel is typically a placeholder for a 3D model that is available elsewhere (e.g. an externally defined ABREP). This GeometricModel contains as minimum a single AxisPlacement that typically represents the origin of the 3D GeometricCoordinateSpace (at point 0/0/0 with no rotation). The G2TMAsso references this AxisPlacement as its Origin attribute and with the Target attribute it refers to a VertexPoint of the EBTRepWLC.

Unlike rigid components, cables and wires are typically flexible and are available at arbitrary lengths with with a (semi-) constant cross-section. Because of this cables and wires have typically no fixed 3D model. However they have typically a constant cross-section, at least within some limits. This 2D geometric cross-section is represented by a 2D GeometricModel. In most cases this GeometricModel contains no details and so consists only of a single 2D AxisPlacement that represents the centre (typically at point 0/0 with no rotation). Like with rigid parts the G2TMAsso references this AxisPlacement with its Origin attribute, but with the Target attribute the G2TMAsso refers to a **Path** of the EBTRepWLC. This Path with its underlying **EdgeBoundedCurveWithLengths** (EdgeBCWL) defines the "Placement" of the wire or cable occurrence within the topological model. In the case that a wire or cable is not round and there is a need to represent the real geometric cross-section this can be done by this 2D GeometricModel as well. For the XML representation this has to be realized by using the subtype ExternalGeometricModel in 2D.

Note: To control to orientation of the flexible cable or wire in 3D it is possible to define a second AxisPlacement that can then be used to be associated with the directrix of a curve_swept_solid_shape_representation (see ISO 10303-523).

Note: This standard does not only support the geometric cross-section of e.g. cable or wire, but also supports the topological cross-section structure of e.g. complex cables or HarnessSegments with their inner structure, see clause 13.

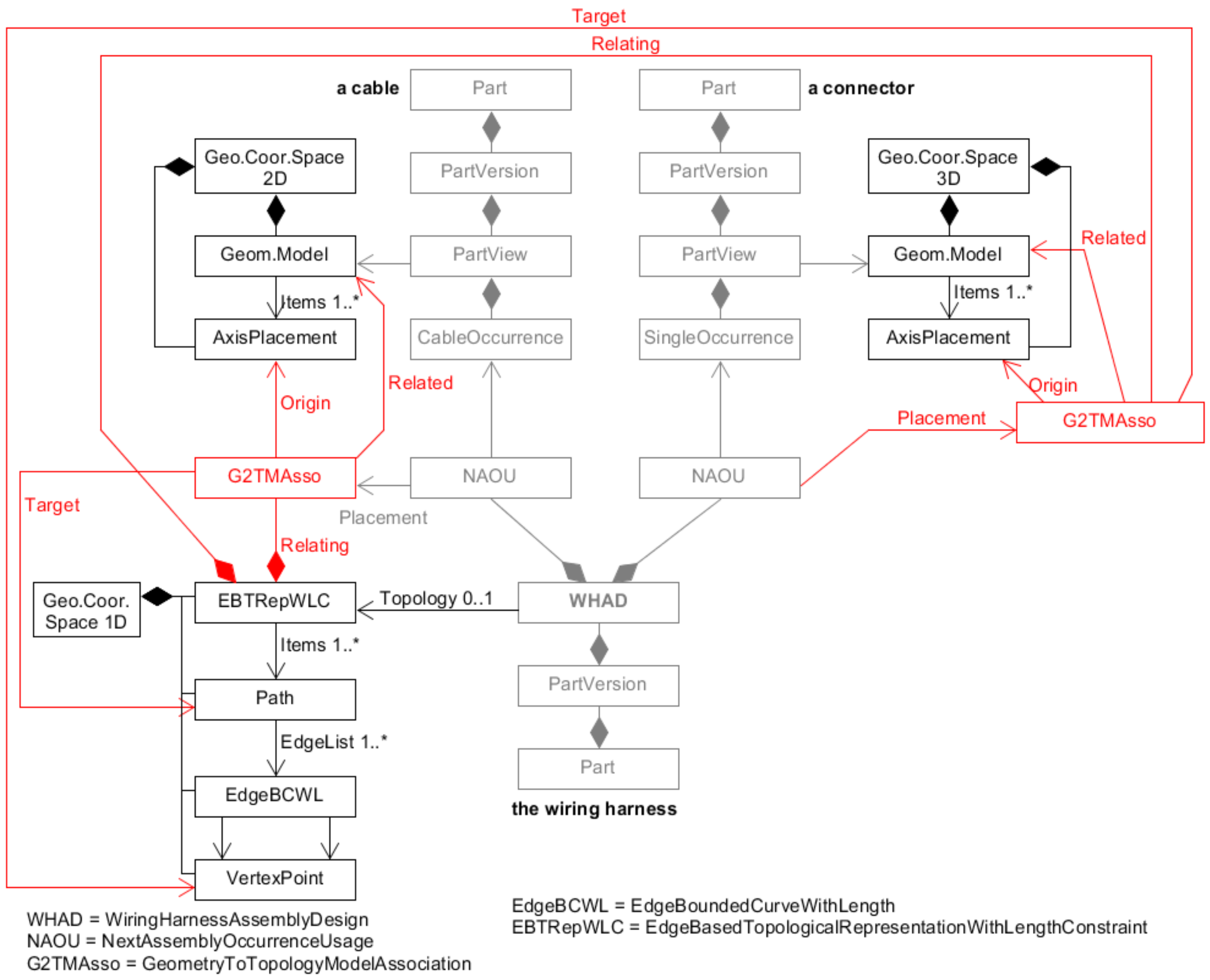


Figure 30: Associating of flexible and rigid part occurrences to the topological wire harness model

Example:

The mapping of flexible objects such as wires, cables and tubular covers to the topological model is done as follows.

1) A generic *GeometricModel* (example _104890) is representing an unspecific 2D cross-section for arbitrary *PartViews* for wires, cables and others. A single *RepresentationItem* of type *AxisPlacement* is defined for the

centre of the cross-section. This *AxisPlacement* is later associated to a topological Path of the topological harness representation and then further on to a curve in a 3D GeometricModel.

```
<!--Generic single AxisPlacement representation for all wire, cable, covering  -->
<RepresentationContext xsi:type="n0:GeometricCoordinateSpace" uid="_104891">
  <Id id="/NULL"/>
  <Representations>
    <Representation xsi:type="n0:GeometricModel" uid="_104890">
      <Id id="xxx"/>
      <Items>
        <RepresentationItem uidRef="_104896"/>
      </Items>
    </Representation>
  </Representations>
  <Items>
  <RepresentationItem xsi:type="n0:AxisPlacement" uid="_104896">
    <Position>0.0 0.0 0.0</Position>
  </RepresentationItem>
  </Items>
  <DimensionCount>3</DimensionCount>
</RepresentationContext>
```

2) A *PartView* refers to the generic geometric cross-section representation as its *DefiningGeometry*. As a consequence this geometric cross-section applies also for the *Occurrences* of the *PartView* (in the example these are wire2/_201104 and wire3/_201204).

```
<Part uid="_101000">
  ...
  <Versions>
    <PartVersion uid="_101001"> ...
      <Views>
        <PartView uid="_101002">
          <DefiningGeometry uidRef="_104890"/> ...
          <Occurrence xsi:type="n0:WireOccurrence" uid="_201104">
            <Id id="wire2"/> ...
          </Occurrence>
          <Occurrence xsi:type="n0:WireOccurrence" uid="_201204">
            <Id id="wire3"/> ...
          </Occurrence>
        </PartView>
      </Views>
    </PartVersion>
  </Versions>
</Part>
```

3) The *WiringHarnessAssemblyDesign* (Part_H1/_311002) contains *NextAssemblyOccurrenceUsage*s (wire2/_315012, wire3/_315041, ...) with *Placement* attribute to *GeometryToTopologyModelAssociation* (subtypes _321080, _321082) and a *NextAssemblyOccurrenceUsage*s (lug1/_315041) with *Placement* attribute to *GeometricRepresentationRelationship* specialization *GeometricRepresentationRelationshipWithPlacementTransformation*.

```
<PartView xsi:type="n0:WiringHarnessAssemblyDesign" uid="_311002">
  ...
  <DefiningGeometry uidRef="_314090"/>
  ...
  <ViewOccurrenceRelationship uid="_315012" xsi:type="n0:NextAssemblyOccurrenceUsage">
    <Related uidRef="_201104"/> <!--WireOccurrence wire2-->
    <RelationType>
      <ClassString>next assembly occurrence</ClassString>
    </RelationType>
    <Placement>
      <GeometryToTopologyModelAssociation uidRef="_321081"/>
    </Placement>
  </ViewOccurrenceRelationship>
  <ViewOccurrenceRelationship uid="_315013" xsi:type="n0:NextAssemblyOccurrenceUsage">
    <Related uidRef="_201204"/> <!--WireOccurrence wire3-->
    <RelationType>
      <ClassString>next assembly occurrence</ClassString>
    </RelationType>
    <Placement>
      <GeometryToTopologyModelAssociation uidRef="_321082"/>
    </Placement>
  </ViewOccurrenceRelationship>
   ...
  <ViewOccurrenceRelationship uid="_315041" xsi:type="n0:NextAssemblyOccurrenceUsage">
    <Related uidRef="_203005"/> <!--SingleOccurrence lug1 -->
    <RelationType>
      <ClassString>next assembly occurrence</ClassString>
    </RelationType>
    <Placement>
      <GeometricRepresentationRelationship uidRef="_314210"/>
    </Placement>
  </ViewOccurrenceRelationship>
   ...
  <Topology uidRef="_321010" />
</PartView>
```

4) A *EdgeBasedTopologicalRepresentationWithLengthConstraint* is representing the topology of a wiring harness. It contains *RepresentationRelationships* that are referenced by the *NextAssemblyOccurrenceUsages* of the wire harness. The specific subtype *GeometryToTopologyModelAssociation* is used to indicate how the flexible centreline of the extruded parts is mapped to either a single *Edge* or a *Path* that is a sequence of *Edges*.

```
<RepresentationContext xsi:type="n0:GeometricCoordinateSpace" uid="_321000">
  <Id id="H1.x Harness topology context"/>
  <Units> ... </Units>
  <Representations>
    <Representation
xsi:type="n0:EdgeBasedTopologicalRepresentationWithLengthConstraint" uid="_321010">
      <Id id="Topological representation of H1 harness"/>
      <Items> ... </Items>
      <!--Transformation of wire2-->
      <RepresentationRelationship xsi:type="n0:GeometryToTopologyModelAssociation"
uid="_321081">
        <Definitional>false</Definitional>
        <Related uidRef="_104890"/>
        <Origin>
          <AxisPlacement uidRef="_104896"/>
        </Origin>
        <Target>
          <Edge uidRef="_321022"/> <!-- Edge S2 -->
        </Target>
      </RepresentationRelationship>
      <!--Transformation of wire3-->
      <RepresentationRelationship xsi:type="n0:GeometryToTopologyModelAssociation"
uid="_321082">
        <Definitional>false</Definitional>
        <Related uidRef="_104890"/>
        <Origin>
          <AxisPlacement uidRef="_104896"/>
        </Origin>
        <Target>
          <Path uidRef="_321066"/> <!-- Path S3-S5 -->
        </Target>
      </RepresentationRelationship>
       ...
    </Representation>
     ...
  </Representations>
  <DimensionCount>1</DimensionCount>
</RepresentationContext>
```

15.5. Association of harness topology to Geometry Models

This clause focuses on the use of the application object TopologyToGeometryModelAssociation. It is used to associate topological items from the `EdgeBasedTopologicalRepresentationWithLengthConstraint` to corresponding geometric representation items in an external geometric model. ... pairwise ...

The geometric model of *WiringHarnessAssemblyDesign* is often represented as a *CompositeGeometrcModel*, that is composed of the flexible part of the harness (harness segments) and the rigid parts such as connectors (at harness nodes). For the representation of rigid parts and there geometries see the CAX-IF recommended practises. For the structure of harness segements some CAX-Systems

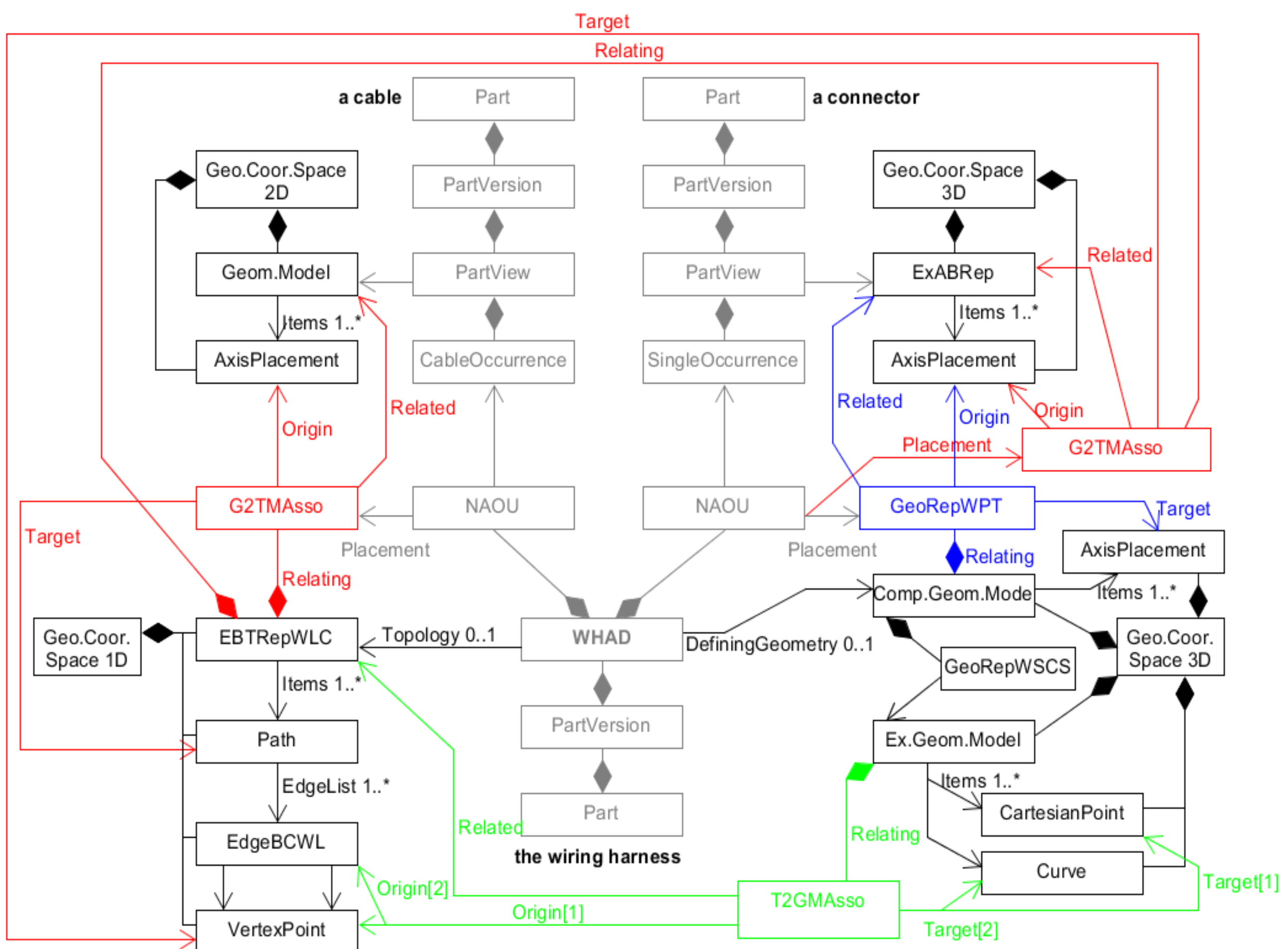
other geometric representations; e.g. the ones for the representation of the flexible harness segments and the ones for the rigid connectors. For the rigid components such as connectors, their geometry is typically brought into the *CompositeGeometrcModel* by a GeometricRepresentationRelationshipWithPlacementTransformation. For the harness segments some CAD system prefer to introduce an artificial product for which there is no place in AP242, as there is no way to build or buy the harness segments without all the rigid parts such as connectors (we can buy individual connectors, but we typically can not buy harness segments). However even these system provide an external geometric model that is used for the *CompositeGeometrcModel* of the overall harness.

There is a ExternalGeometricModel that represents the centre-line curves for the for the topological edges, paths and vertices of the harness model. Also there are external geometric models for the rigid components. multi branchable

Recommendation: To use anchors in p21 files for all the sources and targets (curves, cartesian_points, axis_placements). This capability is available from ISO 10303-21:2016 / edition3 on.

Recommendation: For the external 2D or 3D representation of the centre line curves for the harness segments it is best to use ExternalGeometricallyBoundedWireframeShapeRepresentation or ExternalEdgeBasedWireframeShapeRepresentation for the 3D case on the XML level. These correspond for the p21 level to geometrically_bounded_wireframe_shape_representation or edge_based_wireframe_shape_representation respectively. For the 2D case geometrically_bounded_2d_wireframe_representation should be used on the p21 level when available.

On the right side of figure 31 we see a placement transformation of a rigid part into the harness assembly. The structure is identical to what is detailed in the Recommended Practices for AP242 Business Object Model XML Assembly Structure. E.g. the *PartView* of a connector calls out an *ExternalAdvancedBrepShapeRepresentation* as its *DefiningGeometry*. This representation is defined within a *GeometricCoordinateSpace* (Geo.Coor.Space) and contains an *AxisPlacement* (usually at the position 0/0/0 with no rotation) for placement purposes.



WHAD = WiringHarnessAssemblyDesign
NAOU = NextAssemblyOccurrenceUsage
G2TMAsso = GeometryToTopologyModelAssociation
T2GMAsso = TopologyToGeometryModelAssociation

EdgeBCWL = EdgeBoundedCurveWithLength
ExABRep = ExternalAdvancedBrepShapeRepresentation
EBTRepWLC = EdgeBasedTopologicalRepresentationWithLengthConstraint
GeoRepWPT = GeometricRepresentationRelationshipWithPlacementTransformation
GeoRepWSCS = GeometricRepresentationRelationshipWithSameCoordinateSpace

Figure 31: Flexible (by length) and rigid (2D/3D) model association

A *SingleOccurrence* of the *PartView* of the connector is brought into the *WiringHarnessAssemblyDesign* by a *NextAssemblyOccurrenceUsage*. The *DefiningGeometry* of a *WiringHarnessAssemblyDesign* is represented by another *GeometricCoordinateSpace* that also contains several *AxisPlacements* that are on the positions and orientations on where to place the SingleOccurrences. A *GeometricRepresentationRelationshipWithPlacementTransformation* (GeoRepWPT) brings the ABREP of the connector into the *ComposedGeometricModel* of the *WiringHarnessAssemblyDesign*. The transformation is defined by aligning two *AxisPlacements* onto each other. The *Placement* attribute of the *NextAssemblyOccurrenceUsage* refers to this GRRWPT. This is needed to be able to distinguish the geometry of several occurrences of the same type.

On the left side of figure 31 we see the transformation of flexible parts such as cables and wires into the wiring harness assembly that is different, but similar to the one of a rigid part. As a first step the occurrence of a cable is transformed onto a *Path* or *Edge* of the *EdgeBasedTopologicalRepresentationWithLengthConstraint* that defines the *Topology* of the *WiringHarnessAssemblyDesign* by a *GeometryToTopologyModelAssociation* (G2TAsso). Here the position of the *AxisPlacement* represents the start of the cable that is mapped onto the start *Vertex* of the *Path*. The z-direction (*Axis[3]*) of the *AxisPlacement* represents the direction of the centreline of the cable. As the target *Path* and underlying *Edges* have no particular geometry defined yet, we only know that this direction refers to the direction of the first *Edge* of the *Path* at the start *Vertex*.

At a next step the topological elements of the *EdgeBasedTopologicalRepresentationWithLengthConstraint* are associated to items of the *ExternalAdvancedBrepShapeRepresentation* that defines one of possible many representations of the wire harness. For this a *GeometryToTopologyModelAssociation* (T2GAMP) is used that allows a pair-wise transformation of the items of the *EdgeBasedTopologicalRepresentationWithLengthConstraint* onto the items of a *GeometricModel*. Usually *Edges and Paths* are transformed onto *Curves* and *Vertices* onto *AxisPlacements*. The figure shows a single *Edge* that is mapped onto an external *Curve* within the *ExternalAdvancedBrepShapeRepresentation*.

Note: In the provided example a *GeometryToTopologyModelAssociation* is only used to map *Edges* and *Paths* onto *Curves*. It could also be used to map *Vertices* to *AxisPlacement*. This possible alternative had not been chosen because traditional 3D STEP implementations are used to directly map the 3D model of the piece-parts to the 3D model of the assembly.

1) First we have to identify the STEP p21 file that contains the real Advanced boundary representation of the complete H1 harness. This STEP file might be included in the zip file that also contains the example XML file that is described here.

```
<!--3D geometric model for the Harness design-->
<File xsi:type="n0:DigitalFile" uid="_314080">
  <FileFormat uidRef="_100300"/>
  <FileLocations>
    <FileLocationIdentification uid="_314081">
      <SourceId>H1_abrep.stp</SourceId>
      <SourceType>file</SourceType>
    </FileLocationIdentification>
  </FileLocations>
</File>
```

2) Next we have to create placeholders for the content we expect to find in the external p21 STEP file. For this we create a special RepresentationContext of type GeometricCoordinateSpace that includes the relevant RepresentationItems (here 2 Curves and an AxisPlacement) and an ExternalAdvancedBrepShapeRepresentation:

```
<RepresentationContext xsi:type="n0:GeometricCoordinateSpace" uid="_314091">
  <Id id="/NULL"/>
  <Representations>
    <Representation xsi:type="n0:ExternalGeometricModel" uid="_314090">
                            <Id id="xxx"/>
                            <Items>
                                  <RepresentationItem uidRef="_314092"/>
                                  <RepresentationItem uidRef="_314096"/>
                            </Items>
                            <!--Transformation of topology model-->
                            <RepresentationRelationship
xsi:type="n0:TopologyToGeometryModelAssociation" uid="_314200">
                                  <Definitional>false</Definitional>
                                  <Related uidRef="_321010"/>
                                  <Origin>
                                        <Vertex uidRef="_321041"/>
                                        <Edge uidRef="_321021"/>
                                  </Origin>
                                  <Target>
                                        <AxisPlacement uidRef="_314096"/>
                                        <Curve uidRef="_314092"/>
                                  </Target>
                            </RepresentationRelationship>
                            <!--Transformation of lug1-->
                            <RepresentationRelationship
xsi:type="n0:GeometricRepresentationRelationshipWithPlacementTransformation"
uid="_314210">
                                  <Definitional>true</Definitional>
                                  <Related uidRef="_103090"/>
                                  <Origin uidRef="_103096"/>
                                  <Target uidRef="_314096"/>
```

```
                                </RepresentationRelationship>
                                <ExternalFile uidRef="_314080"/>
<!-- specific #15=SHAPE_REPRESENTATION -->
    </Representation>
  </Representations>
  <Items>
    <RepresentationItem xsi:type="n0:Curve" uid="_314092">
      <External xsi:type="n0:NextInstanceInverse" uid="_314092_1">
        <Id id="#15"/>
<!-- #15=SHAPE_REPRESENTATION(' ',
(#1917,#5143,#31773,#44222,#45076,#45089,#45943,#47475,#50537,#50845,#51147,#51449,#517
51,#51772),#21) ; -->
        <AttributeName>rep_2</AttributeName>
        <NextInstance  xsi:type="n0:NextInstanceForward" uid="_314092_2">
          <Id id="#1915"/>
<!-- #1915=(REPRESENTATION_RELATIONSHIP(' ','
',#24,#15)REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#1916)SHAPE_REPRESENTATION_RE
LATIONSHIP()) ; -->
          <AttributeName>rep_1</AttributeName>
          <NextInstance  xsi:type="n0:NextInstanceForward" uid="_314092_3">
            <Id id="#24"/>
<!-- #24=SHAPE_REPRESENTATION(' ',(#1918),#23) ; -->
            <AttributeName>context_of_items</AttributeName>
            <NextInstance  xsi:type="n0:NextInstanceInverse" uid="_314092_4">
              <Id id="#23"/>
<!--
#23=(GEOMETRIC_REPRESENTATION_CONTEXT(3)GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT((#22))GLOBA
L_UNIT_ASSIGNED_CONTEXT((#17,#18,#19))REPRESENTATION_CONTEXT(' ',' ')) ; -->
              <AttributeName>context_of_items</AttributeName>
              <NextInstance  xsi:type="n0:NextInstanceForward" uid="_314092_5">
                <Id id="#96"/>
<!-- #96=GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION('NONE',(#97),#23) ; -->
                <AttributeName>items</AttributeName>
                <NextInstance  xsi:type="n0:NextInstanceForward" uid="_314092_6">
                  <Id id="#97"/>
<!-- #97=GEOMETRIC_SET('NONE',(#90,#99,#104,#109,#114,#119,#124,#141,#157)) ; -->
                  <AttributeName>elements</AttributeName>
                  <NextInstance  xsi:type="n0:ExternalEntityInstance" uid="_314092_7">
                    <Id id="#141"/>
<!-- #141=COMPOSITE_CURVE('Flexible Curve.2',(#140),.U.) ; -->
                  </NextInstance>
                </NextInstance>
              </NextInstance>
            </NextInstance>
          </NextInstance>
        </NextInstance>
      </External>
    </RepresentationItem>
    <RepresentationItem xsi:type="n0:AxisPlacement" uid="_314096">
      <External uid="_314096_1">
        <Id id="#5143"/>
      </External>
      <Position>0.0 0.0 0.0</Position>
    </RepresentationItem>
    </Items>
    <DimensionCount>3</DimensionCount>
</RepresentationContext>
```

The *ExternalAdvancedBrepShapeRepresentation* contains a *GeometryToTopologyModelAssociation* that maps two origen objects (a Vertex and an Edge) of the

EdgeBasedTopologicalRepresentationWithLengthConstraint onto corresponding target objects (here *AxisPlacement* and curve) of the complete H1 geometry.

There is also an *GeometricRepresentationRelationshipWithPlacementTransformation* that maps the defining geometry the terminal lug (here a ExternalAdvancedBrepShapeRepresentation) to a corresponding position and orientation in the H1 geometry.
For further details see CAx-IF Recommended Practices AP242 BO Model XML Assembly Structure (version 2?)

16. Overall connectivity in top Assembly

SO FAR INCOMPLETE

STEP AP242 ed2 has two ways to express connectivity, either by AssemblyShapeJoint or by PartConnectivityDefinition. How they differ and when to use which one:

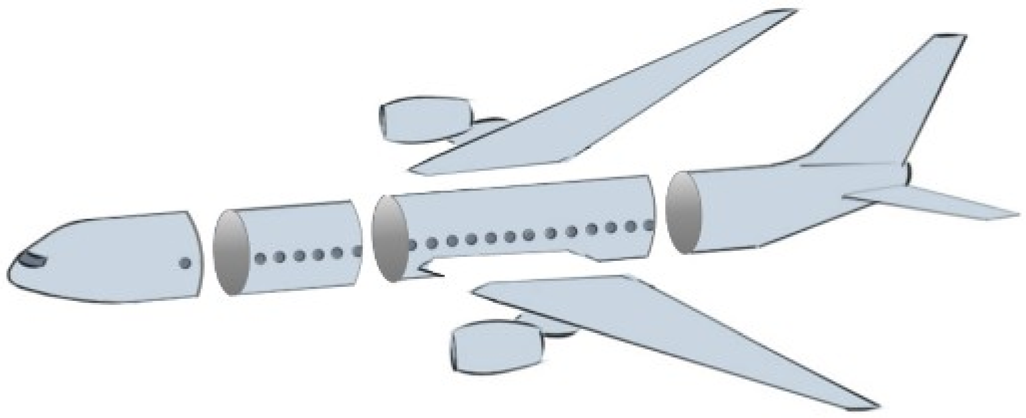
- As the name indicates, an AssemblyShapeJoint can only be used for an AssemblyDefinition (or subtype WiringHarnessAssemblyDesign) that contains Occurrences (aka devices) that are connected with each other. This is done by referencing the two or more OccurrenceShapeFeatures(or subtypes OccurrenceContactFeature and OccurrenceTerminal) of these Occurrences
- An AssemblyShapeJoint is used for the represenation With AssemblyShapeJointItemRelationship two or more any kind of OccurrenceShapeFeatures (e.g. OccurrenceTerminals) can be connected. This might be for pure mechanical reasons (OccurrenceShapeFeatures or OccurrenceContactFeature), but can also be to exchange energy or information (OccurrenceTerminal).
- A PartConnectivityDefinition can be used in different way
  - for a part to tell which terminal are internally connected
  - for an assemble to state which internal devices are connected
  - for an assembly to state how the internal devices are connected by wires, cables, harnesses ... / also covered by AssemblyShapeJoint
  - for an assembly to state how the terminal of internal devices are made accessible to the outside by terminals of the assembly

OccurrenceTerminal,

PartTerminal

poke-home

Some aircrafts (especially bigger ones) are build in sections with each having its own wiring harness(es). E.g. separate sections for the cockpit, the front fuselage, the central fuselage, the rear fuselage, the wings and the turbines. These sections might be manufactured and assembled in different factories. Only in the final assembly line these sections are combined into the final aircraft, and part of this process is that the wiring harnesses in the separate sections are joined by connecting corresponding plugs and receptacles from different harnesses. By this the overall electrical system of an aircraft is set up. For each such join a single AssemblyShapeJoint is needed for the purpose of data exchange.



When all the high level AssemblyShapeJoints between all the plugs and receptacles for the top assembly of a vehicle are given, a receiving system can then deduce all the lower level AssemblyShapeJoints and so generate the overall electrical network of the whole top assembly.

Connectivity between all the devices (sensors, actors, computers, batteries, lamps, ...) in a top assembly.

SO FAR INCOMPLETE