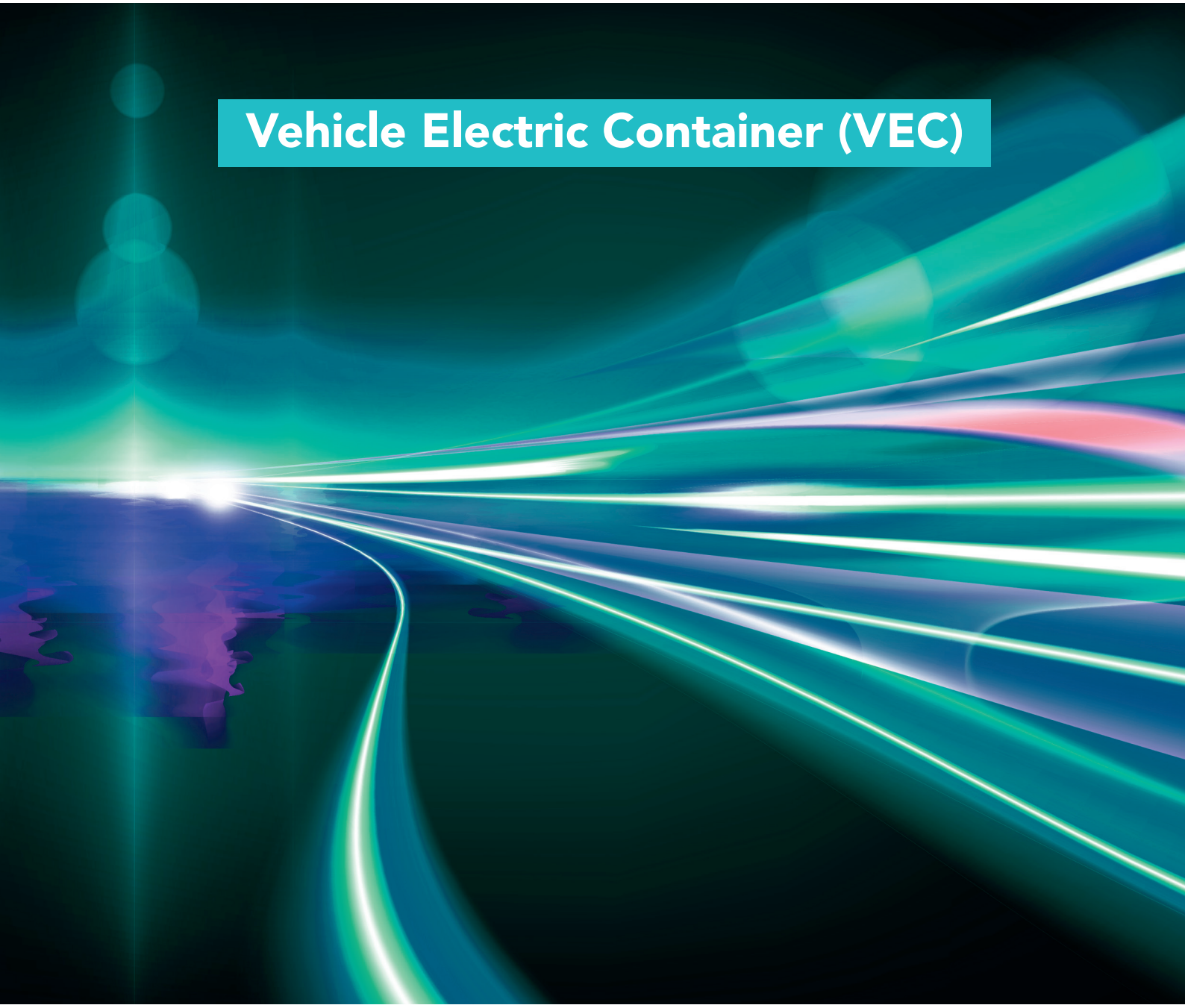
prostep IVIP

VDA | Verband der Automobilindustrie



prostep ivip / VDA Recommendation

Vehicle Electric Container (VEC)

Version 1.2

Disclaimer

VDA / prostep ivip Recommendations (VDA / PSI Recommendations) are recommendations that are available for general use. Anyone using these recommendations is responsible for ensuring that they are used correctly. This VDA / PSI Recommendation gives due consideration to the prevailing state-of-the-art at the time of publication. Anyone using VDA / PSI Recommendations must assume responsibility for his or her actions and acts at her/his own risk. The prostep ivip Association, the VDA and the parties involved in drawing up the VDA / PSI Recommendation assume no liability whatsoever. We request that anyone encountering an error or the possibility of an incorrect interpretation when using the VDA / PSI Recommendation should contact the prostep ivip Association (psiissues@prostep.com) immediately so that any errors can be rectified.

Copyright

Contents

List of figures

1 General

1.1 Preamble

The complexity of today's vehicle electrical systems is constantly growing. A vast variety of options is on the market. Firmly organized and integrated cross-company development processes are essential, combined with powerful, integrated IT infrastructures to support all cross stakeholders.

Against this background, the prostep ivip project group “Vehicle Electrical Systems Workflow Forum” and its predecessors have developed standardised data formats for the uniform description of wiring harnesses and related data. Providing the Harness Description List (KBL, PSI 19/VDA 4964) and supplementing schemas was a leap forward regarding the improvement of car electric development processes and their integration in the development processes for complete vehicles.

However, supporting the whole electric development processes and providing an integrated view on the complete electrical network of a vehicle was not in the scope of the provided specifications. Additional use cases must be addressed. The objective of the prostep ivip project group “Vehicle Electrical Systems Workflow Forum” is to collect these use cases and specify the Vehicle Electric Container (VEC) based on them as the required standardised data format in this context.

The VEC data format specification harmonizes and integrates the already existing solutions with the newly gathered requirements. The VEC data format specification addresses a significantly extended amount of use cases, focussing not only on one single wiring harness but on an electric system as a whole. The VEC data format specification supports a great variety of data exchange use cases all along the electric system development process.

The definition of the VEC was done with a focus on the requirements of the automotive industry. However, it is not restricted to this domain and it is expected that the VEC specification is applicable in aerospace industry and others as well.

1.2 Objectives of the recommendation

This Recommendation contains the specification of the VEC data format with the objective to

- Define a commonly agreed vocabulary and object semantics for the domain of the electrical design process in vehicles
- Facilitate data exchange between development and business partners in the context of physical electric system development and production planning
- Enable tool integration as well as tool-spanning traceability and tool-spanning change management
- Reduce complexity and at the same time increase flexibility by better decoupling tools and data
- Support paperless processes
- Provide a perspective for a solution for requirement in the context of long-term preservation

Concrete use cases are described in chapter 2.

Note: The VEC data format definition is explicitly not intended to be interpreted as a recommendation for the definition of the internal database structure of software tools.

1.3 Changes to preceding versions

Between this Version (1.2) and the direct predecessor (Version 1.1) over 190 individual issues have been addressed. The following section lists the main subjects that have been changed, improved or added. A complete and detailed change history is available in the ECAD Wiki and in the issue tracking system.

Changes that affect the resulting schema in an incompatible way are marked with a “X” in the last column. For more details on compatibility see Chapter 1.4.

Change	Inc.
Reorganization of the Model Outline (Chapter 5)	
Added “General Guidelines” for requirements on VEC implementations that are not strictly related to the model structure (Chapter 4).	
Added model documentation to the generated XML Schema files.	
General orthogonal grouping concept to represent functional mappings and requirements (see AssignmentGroup)	
Added concept for the instantiation of topologies.	
Added concept for hierarchical topologies supporting multiple use case (e.g. better traceability between geometry and harness process, splice position optimization, layered segments with a defined inner structure, composite segments, …)	
Added concept for assigning topologies to zones.	
Completely revised the interpretation of Net- & ConnectionSpecification (Architectural Layer & System Schematic)	X
Refactoring of the multi-core representation	X
Added support for FIT-Rates for components	
Added concept to express conformance with requirements (see RequirementsConformanceSpecification)	
Added concept to define application constraints on instances (e.g. component nodes) (see ApplicationConstraint)	
Added concept for common variant configurations (base inclusion)	
Added concept to define system schematic traceability for directly mated E/E components.	
Added concept to define multicores in their usage (similar to twisted pairs)	
Added concept for traceability between wires and their respective fusing.	

Added concept to define bending restrictions on topologies.	
Added concept to define baselines (well defined sets of ItemVersions)	
Added concept to integrate with the 3D geometries of individual components (e.g. bounding box,	
Added concept for default tolerance definitions	
Added concept for wire addons in connectors.	
Allowed part usage (component instances without part number) in the bill of material.	
Added support for component selection tables.	
Added concepts to support 150% E/E component definitions.	
Added concepts for the description of fuse boxes and other E/E-Components • internal connectivity • variance of internal connectivity • modularity	
Improved modification tracking / change detection for the digital representation of documents (independent from the approval process in the domain)	
Refactored 3D representation of segments. Dropped current 3D-curve model and replaced it by complete representation of NURBS.	X
Added concept for integrated terminals and supplementary components in different contacting situations (e.g. wire fixations)	
Clarification that contact points are free of variance.	
Refactored attributes for compatibility definitions between terminals, plugs, cavities, seals and wires.	X
Added concept for flat band wires and flat cores.	
Dropped support for conformance classes.	X
Added support for grouping component ports by connector.	X
Definition of complex part relations	
Support for complex custom properties und multiple primitive types.	
Added support for hierarchical structures on variant groups and added multiple attributes to the classes in the variant configuration scope.	
Added support for grommets sealed with additional single wire seals.	
Refactored concept for supplementary parts of components in specified locations (e.g. Slots)	X

Added support for diodes	
Added support for cable ties	
Added support for multi-fuses	

The following list contains all minor changes, that affected schema compatibility.

Refactored and renamed “ContactSystem” to TerminalPairing	X
Path mistakenly inherited from ConfigurableElement	X
Moved “referenceElement” Association from PartOccurrence to OccurrenceOrUsage	X
Refactoring of WireProtectionRole, introduction of TapeRole	X
Redefined semantics for ConnectionGroup and NetGroup	X
Removed SealingClass and AbrasionResistanceClass (replaced by general concept RobustnessProperties).	X
Removed CompatibilityStatement & CompatibilitySpecification	X
Refactored modular slot definition (now using indirect references with PartVersion)	X
Refactored CopyrightInformation	X
Moved attribute TerminalSpecification.angle to WireReception	X
Removed Signal from Net-Layer	X
Refactoring of SheetOrChapter	X
Deprecation of CavityDesign in TerminalReceptionSpecification and CavitySpecification	X

1.4 Compatibility to preceding versions

Version 1.2 is an extension of version 1.1. Model changes and extensions are guided by the fundamental principle of keeping already implemented concepts downward compatible as far as possible. However, this was not possible in all cases.

Compatibility is defined in the context of this document as the possibility that XML documents created for version 1.1 are still (schema) valid version 1.2 documents. In that sense, incompatible changes will result in schema validation errors if the version 1.1 file uses the affected model elements. Such changes are listed in Chapter 1.3 explicitly.

Additionally, version 1.2 introduces a large amount of open enumerations. As this reduces the degree of freedom in the model it is very likely that version 1.1 VEC files will not validate against the 1.2 strict schema.

Other changes that might be interpreted as incompatible, even without producing schema validation errors, are all improved or clarified documentations, as it might occur that earlier interpretations are now explicitly invalid model interpretations.

All VEC implementations that currently use custom properties for elements that have now (introduced with this version) explicit concepts should be changed accordingly.

1.5 Document structure

Chapter 2 describes some exemplary use cases for the application of the VEC data format.

Chapter 3 breaks down the requirements that the VEC data format specification has to meet.

Chapter 4 explains general concepts and guidelines that apply in a more cross-sectional way and cannot be linked to a specific individual model element.

Chapter 5 explains the meta model of the VEC data format and explains the concrete XML-based syntax of the VEC data format.

Appendix A contains a glossary for the most common abbreviations.

Appendix B contains the detailed meta model specification with a definition of all classes, attributes and relationships in alphabetical arrangement.

1.6 Abbreviations, terms and definitions

See Appendix A for a list of relevant abbreviations, terms and definitions.

1.7 Reference

Further information about this recommendation and related documents and specifications (e.g. the VEC.xsd) are available from

- The VDA and its working party PLM (see http://www.vda.de)
- The prostep ivip Association respectively the project groups VES Workflow Forum and ECAD-implementer forum (see https://www.prostep.org/en/medialibrary/publications, https://ecad-wiki.prostep.org/)

In addition to that, special reference goes to the recommendation PSI 19 / VDA 4964 Harness Description List (KBL) as previous recommendation.

2 Exemplary business use cases

This chapter describes some exemplary business use cases for the application of the VEC data format. The VEC has been designed with these business cases in mind. However, this list shall not be considered as complete or as a restriction of the usage of the VEC.

2.1 UC1: Exchange of components data (part master data)

The VEC data format specification must have a scalable concept for the exchange of part master data. This includes at least the following sub use cases which can appear separately or in any combination:

- UC1.1: Exchange of components data (part master data) with focus on their technical features
    - Example: components data exchange between components supplier and OEM in order to support design engineers to find and choose applicable parts out of a technical perspective
- UC1.2: Exchange of components data (part master data) with focus on meta data
    - Example: components data exchange between components supplier and OEM in order to support design engineers to find and choose applicable parts out of an organisational perspective (e.g. considering approval information, existence of certain usage constraints, …)
- UC1.3: Exchange of components data (part master data) with focus on relational data
    - Example: components data exchange between OEM and a development partner for the definition which cavities, terminals, cavity seals, cavity plugs, and wires are compatible respectively approved
    - Example: Supply tools with data that produce the final wiring harness definition (in combination with a geometry and a connectivity specification as well as inclusive steps like automatic terminal calculation)
- UC1.4: Exchange of components data (part master data) as far as needed for the understanding of a wiring harness description
    - Example: The KBL case

2.2 UC2: Exchange of connectivity data

The VEC data format specification must have a scalable concept for the exchange of (normally configuration based) connectivity data in the context of the car electric system. This includes at least the following sub use cases which can appear separately or in any combination:

- UC2.1: Exchange of architectural data
    - Example: For seamless traceability up to an abstract function level
    - Example: Abstract description of necessary system connections
    - Example: Supply schematics design tools with data in order to facilitate the schematics design process

- UC2.2: Exchange of schematics data
    - Example: Exchange of schematics data between tools for concept tools (tools that allow the evaluation of different electric architectures) and conventional schematics design tools.
    - Example: Supply schematics design tools with data in order to facilitate the wiring design process
    - Example: Supply routing tools with data (in combination with a geometry specification)
- UC2.3 Exchange of wiring data
    - Example: Support data exchange between OEM and a development partner that can be either requested to complete a wiring definition or alternatively provide a final wiring harness definition
    - Example: Supply tools with data that produce the final wiring harness definition (in combination with the necessary components data and a geometry specification as well as inclusive steps like automatic terminal calculation)
    - Example: Supply routing tools with data (in combination with a geometry specification)
    - Example: Supply tools with data that support the tracing of energy paths
    - Example: Supply tools with data that support the service documentation process

2.3 UC3: Exchange of geometry data

The VEC data format specification must have a scalable concept for the exchange of (normally configuration based) geometry data in the context of the car electric system. This includes at least the following sub use cases which can appear separately or in any combination:

- UC3.1: Exchange of topology and relating part placement information
    - Example: Support data exchange between OEM and a development partner who can be either requested to complete a geometry description or alternatively provide a final wiring harness definition
    - Example: Supply tools with data that produce the final wiring harness definition (in combination with the necessary components data and a connectivity specification as well as inclusive steps like automatic terminal calculation)
    - Example: Supply routing tools with data (in combination with a connectivity specification)
    - Example: Supply tools with data that support the service documentation process
    - Example: Supply concept tools with geometry data (tools that allow the evaluation of different electric architectures)
- UC3.2: Exchange of 2D harness drawing data

- Example: Support layout-based discussions between OEM and a development partner
- Example: Supply tools with data that have to visualise topology data

- UC3.3: Exchange of 3D wiring harness geometry data
  - Example: Support data exchange between OEM and a development partner respectively harness manufacturer
  - Example: Supply 2D harness drawing tools with data

2.4 UC4: Exchange of wiring harness data

The VEC data format specification must have a scalable concept for the exchange of (normally configuration dependent) wiring harness data. This includes

- The relevant part master data (compare UC1.4)
- The relevant wiring data (compare UC2.3)
- The relevant geometry data (compare UC3.1 - UC3.3)

At least the following sub-use cases have to be supported:

- UC4.1: Exchange of bill of material data (e.g. module-based)
  - Example: Supply tool for cost and/or weight calculation with data
- UC4.2: Exchange of connection list data (e.g. module-based)
  - Example: Supply tools for test application development with data
- UC4.3: Exchange of complete 150% wiring harness data
  - Example: Support data exchange between OEM and harness manufacturer in the context of the processes cost calculation, production planning and change management
  - Example: Supply reporting tools with data
  - Example: Supply EMC-simulation tools with data

2.5 UC5: Exchange of an integrated model of a complete vehicle network

The VEC data format specification must offer a concept for the data exchange of the above-mentioned use cases in an integrated way. This includes traceability links between the different sections of the model.

- UC5.1: Exchange of all wiring harnesses of a vehicle together.
  - Example: Supply customer service diagnosis with relevant information to localize failures.
  - Example: Perform simulations related to the complete vehicle network (e.g. voltage drops)
- UC5.2: Exchange of wiring harness data together with its different geometrical views (e.g. 3D mounting position, form board)
  - Example: Generate 100% variants of the harness geometry to support collision detection in the DMU.

    - Example: Support the inclusion of environment aspects (e.g. heat, crash) in automated validation algorithms.
- UC5.3: Exchange of wiring harness data together with its corresponding system schematic.
    - Example: Supply simulations tools (e.g. EMC) with information about signals and system relevance of concrete wires.
    - Example: Supply engineers and workshop personal with information to allow a better understanding of the role of specific harness elements in the context of the overall system.

3 Requirements on the VEC

This chapter summarises the requirements for the VEC data format.

3.1 Requirements in the PDM context / addressing PDM data

3.1.1 Project reference

Each part / part version description (not part instance description) must be able to express a project reference respectively a special model series reference. The reference is meant to specify the project respectively the model series for which the part has been / is going to be developed.

3.1.2 Part Usage Constraints

Each part / part version description must be able to optionally express one or more dedicated constraints about the permitted usage of that part. These constraints are meant to be independent of approval information and can address the following aspects

- Vehicle
    - Vehicle project / model series (CarClassificationLevel2)
    - Vehicle derivative (CarClassificationLevel3)
    - Vehicle model (CarClassificationLevel4)
- Technical Equipment (by option code, e.g. country code, for special-purpose vehicles)
- Time constraints
    - Absolute time constraints
    - Constraints addressing the construction phase / model year
    - Constraints addressing the serial number
- Point of use in the context of the electric system

3.1.3 Approval information (stating a general permission for use)

For each item / item version (item: part or document), the VEC data format specification must have a concept to optionally express the status of approval. The following values are cross-company agreed:

- NotYetApproved
- Approved
- Withdrawn

Beyond that, the VEC data format specification must have a concept to cope with company specific values concerning the approval level. Finally, the VEC data format specification must have a concept to express further details of a certain approval

- A certain identification
- The person (name, company and department) who approved the part
- The date of approval

3.1.4 Change history

For each item / item version (item: part or document), the VEC data format specification must be able to express the complete change history. This addresses the following aspects

- Creation information
    - The person (name, company and department) who created the item
    - The creation date
- Version / Version history information
    - The version index of each item version
    - The direct predecessor item versions (by item number and item version) regarding the two agreed item version types “Derivation” and “Sequence”
- Change information
    - The one or more change descriptions each specifying one or more of the following aspects
        - The description of an actual change in comparison to the direct predecessor-version(s) in an informal way
        - A certain label (relevant for drawings)
        - The change date
        - The responsible designer
        - The related change order
        - The related change requests
        - The approver of the change

Note: The VEC data format specification must have a concept to express change history information as specified completely in separate which means without any further technical details.

3.1.5 Conformance with Requirements

The VEC data format specification must have a concept to express conformity (or non-conformity) of parts with certain requirements or specifications. There are various use cases where parts / components are qualified against specific requirements or specifications (e.g. type-examinations, suitability for certain areas of application or manufacturing methods). The result is an explicit statement about the conformity (or non-conformity) of the component.

3.1.6 External References

For each part the VEC must be able to specify a document reference (by referring document number and version). Normally, such references can be resolved over a period of several years (which is better than a combination of a file path and file name).

In addition, for selective content/elements the VEC must be able to explicitly refer to external files. This can be helpful e.g. for files containing graphically represented installation instructions.

3.1.7 VEC file-spanning correlations

The VEC data format specification must have a transparent concept that makes it easy for analysing tools to unambiguously calculate correlations between one VEC file (received at a time A) and another VEC file (received at a different time B). Thereby, the following types of correlation are required to be recognisable:

- No correlation
- Extension of information
- Changes

3.1.8 Baselines

The VEC data format specification must have a concept to group versioned elements (parts & documents) to a valid baseline.

3.2 Requirements addressing variance information

A multi-variant system/product is often described by a set of variant-free (non-variant) building blocks and a set of rules that defines under which conditions these building blocks are contained in a manufactured product. Such a specification is often referred to as a 150%-definition, because it contains more features than any actually manufactured product. The VEC data format shall contain concepts to support this approach in an appropriate way.

To express 150%-definitions it is required to specify variant configuration terms on relevant configurable elements. The variant configuration terms express the conditions for a product configuration, under which the respective element exists. Depending on the process, this configuration mechanism can be attached to different design elements (e.g. connections in a system schematic, topology segments in a harness geometry definition or modules in a customer specific harness (KSK).

Variant configuration terms are based on a set of product properties/features with cross relations (e.g. mutual exclusion, dependency on each other). Together they define the possible variety of the product. The VEC shall not restrict the amount, the naming nor the semantics of these properties/features and the cross relations. Examples for such properties/features are: the product model, the right- or left-hand-driving, the engine fuel type, the body style as well as different equipment features such as seat heating, air conditioning or driver assistance systems.

The VEC data format specification shall support definition of variant and non-variant elements. Non-variant elements are characterized by the fact that their content is either completely or not at all contained in a manufactured product (e.g. a harness module in a customer specific harness, or an atomic component like a connector). Non-variant elements can in turn be part of variant elements. Variant elements are characterized by the fact that their content is dependent on a concrete product configuration and normally only a fraction of the content is contained in a manufactured product (e.g. the description of a customer specific harness).

The following chapters specify some further general requirements concerning the expression of variance information. In addition, the subchapters of the physical harness, component description, topology/geometry, and connectivity perspectives specify

- For which concrete elements the VEC needs to be able to specify variant configuration terms.
- How and for which elements the VEC needs to be able to express non-variant content respectively variant content.

3.2.1 Variant configuration terms

It is required that the VEC data format specification has a unified concept for the assignment of variant configuration terms. Elements for which the appearance in a configuration is dependent on the same motivation shall be able to refer the same variant configuration term instance.

The VEC data format specification is required to offer two alternative ways for the expression of a variant configuration term which are equivalent out of the standardisation perspective.

- Expression as a character string that meets a standardised syntax that is either directly defined by the VEC data format specification or by related documents.
- Expression as a character string without any limitations by the VEC data format specification.

3.2.2 Inheritance of variant configuration terms

The VEC data format specification must have a concept to define variant configuration terms that are in common for a set of elements and that are extendable for specific elements. E.g. all elements in a system schematic share a common base variant configuration term. However, some elements can have more restrictive variant configuration terms.

3.2.3 Vocabulary of variant configuration terms

The VEC data format specification must have a concept that allows the separate / independent definition of the vocabulary (the set of valid literals) that is used for the specification of variant configuration terms. This concept must be free of constraints concerning the number of identifiers as well as the naming of the identifiers. Furthermore, the concept is required to allow the specification of the complete vocabulary or alternatively a selected fraction.

Note: The VEC data format specification is not wanted to standardise the vocabulary itself respectively any identifiers.

The concept for the specification of the vocabulary must allow the definition of identifiers for elementary elements as well as identifiers for grouping elements. Some examples for elementary elements are: “LL” (left-hand drive vehicle), “CAB” (cabriolet) and “ESC” (electronic stability control). Grouping elements must be able to refer various elementary elements. Thereby, the grouping type must be able to be defined. Some examples for an aggregating grouping are: “series” (definition of the properties/features that are included in the standard configuration) and “winter” (a certain equipment package). An example for an exclusive grouping is the vehicle driving type which can be either right-hand driving or left-hand driving.

3.3 Requirements out of the physical harness perspective

Out of the KBL-perspective, the VEC data format specification must enable the description of all data that a manufacturer needs to plan the wiring harness production. This includes the description of single wiring harnesses as well as so called 150%-harness descriptions that may be furthermore structured in several potentially overlapping modules.

3.3.1 Part descriptions

The VEC data format description must have a concept which allows the specification of the relevant part master data. This includes, but is not restricted to:

- Identification
    - Part number
    - Version identification
    - Company reference
- Reference to relevant documents (e.g. drawings, requirements specifications, …)
- Abbreviation (e.g. a certain string that is generated out of certain attributes)
- Description (e.g. “Connector Housing 5-pin”)
- Material together with a reference to the relevant material reference system (e.g. DIN, VDE, …)
- Colour information together with a reference to the relevant colour reference system
- Mass information (together with the corresponding unit)
- FIT rates
- In case of wires
    - Cross section area
    - Wire type
    - In case of multi core wires an identification for each of the single cores
    - Wire groupings (together with the relevant twisting specification)
- In case of connector housings
    - Coding
    - Structure
        - Slots together with an identification for each slot
        - Cavities together with an identification for each cavity
    - Modular connectors
    - Required addons for wires
- In case of terminals
    - Terminal type
    - Nominal size

3.3.2 E/E component interfaces

The VEC format specification must have concept to describe the interfaces of E/E components with information that is required within the scope. This includes, but is not restricted to:

- Mechanical / physical interface
  - Connector geometry
  - Pin properties (e.g. plating material)
- Electrical Interface
  - Voltage level (variant dependant)
  - Current profiles (variant dependant)
  - Internal connectivity (variant dependant, but not software controlled)
- Pluggability
  - with the wiring harness
  - between E/E components (e.g. fuse box & fuse)

3.3.3 Assembly / harness module / harness configuration set parts list

The VEC format specification must have a uniform concept for the unambiguous specification of parts lists (addressing all associated subparts like connector housings, terminals, wires optionally together with the length information for each core, wire protections, fixings, grommets, and their quantity). This is equally relevant for simple assemblies as well as harness modules and complete harness configuration sets. The parts lists have to be version precise.

3.3.4 Wire list and connection list

For each wire instance (KBL::Wire_occurrence, KBL::Core_occurrence), the VEC data format specification must allow the definition of

- The relevant part master data both for the whole wire and for each core
- The wire instance number
- The contact points the cores interconnect (regarding the contacted pluggable terminal, splice terminal or ring terminal, optionally in combination with a contacted cavity)
- The length information for each core together with a length type (e.g. the DMU-length)
- The connections (KBL::connection) the cores realise
- A signal name for each core respectively connection

3.3.5 Cavity equipment

The VEC concept for 150% wiring harness descriptions must ensure that for concrete configurations the cavity equipment for each cavity of the included connectors can be unambiguously calculated. This addresses the following information

- What cavities are equipped with which terminal(s) and optionally together with which cavity seals. Which wires / cores are connected?

- What cavities are non-plugged or plugged and, in this case, plugged with which plugs?

3.3.6 Replacements

The VEC data format specification must have a concept which allows the definition of configuration dependant cavity plug replacements. The cavity plug replacement is e.g. needed in the case that a certain configuration results in a certain combination of modules where at least one module defines an equipment for a cavity that belongs to a plugged connector which is part of another module.

3.3.7 Topology and Routing

The VEC data format specification must have a concept that allows the definition of the harness topology. Furthermore, for each connection (KBL::connection) or wire instance (KBL::Wire_occurrence, KBL::Core_occurrence) it must be possible to define a routing which means an obligatory path through the topology. Comparably, it must be possible to specify the placement of all other part instances (fixings, grommets, …) as a defined location on the topology. The VEC must support components that are attached to multiple locations in the topology (e.g. cable ducts, grommets, connectors).

The routing definition concept must offer the possibility to specify the segments that are mandatory even in the case of a calculation of a new routing. In addition to that, the routing concept must offer the possibility to mark certain routing definitions as special. This is intended to offer for instance an optional marking mechanism to enable the distinguishing of man-made routing definitions and calculated ones.

The VEC must provide all information necessary for routing algorithm to create an automatic routing to and from connectors with multiple bundle position points (e.g. HV-connectors). This includes the information which cavity is reachable from which bundle position point.

3.3.8 Wire and core lengths

The VEC data format specification must have a concept which allows for each wire and core instance the definition of dedicated lengths information each of it in combination with the relevant length type (e.g. the calculated DMU length, a measured value, …)

3.3.9 Modules and Harness configuration sets

The VEC data format specification must have a concept that allows the definition of modules (KBL::Module) and complete harness configuration sets (KBL::Harness_configuration) on the basis of a common 150% wiring harness description. Thereby, the definition of harness configuration sets must be alternatively able to base on the aggregation of elementary part instances (like connector housings) or modules or a combination of both.

Modules and Harness configuration sets have parts character which means they have a part number, version index and optionally all kinds of PDM information like any other part.

3.3.10 Extra requirements addressing variance information

The VEC data format specification must have a concept that enables the assignment of every occurrence, every module (instance) and every harness configuration set

(instance) in the context of a 150% wiring harness description with a certain variant configuration term.

Furthermore, the VEC data format specification must have a concept to describe mutually exclusive modules as part of a module family (KBL – module family concept).

Finally, the VEC data format specification must have a concept to describe certain occurrences as mandatory completion parts in dependency of the existence of one or more related modules (KBL - module list concept).

3.3.11 Definition and reuse of complex parts

The VEC data format specification must have a concept to define complex parts and reuse them in different positions. Wiring harnesses do not exclusively consist of simple components, but also of parts that are assembled themselves. This can range from ready-made cables (e.g. antenna or USB cables) up to complete (sub) harnesses (e.g. automatically manufactured modules or door harnesses used in the left and right door in the same manufactured way).

The reuse concept shall support at least the following aspects:

- Renaming in the context of usage (e.g. connector names, wire numbers)
- Mapping into the topology / geometry of the usage (e.g. right door / left door)
- Correct electrological mapping in the context of the usage (e.g. realized schematic connections & signals are usage dependant).
- Redefinition of properties in the context of the usage (e.g. some ready-made cables with connectors on only one side might have a fixed length and are cut to the correct length in the concrete usage).
- Definition of the installation in the concrete usage (e.g. positioning of connectors, routing of cables)

3.4 Additional requirements out of the component description perspective

3.4.1 Usage Nodes

The VEC must provide a concept to support usage nodes. Usage nodes are a representation of an abstract position in a vehicle, independent from the concrete perspective (physical harness, topology/geometry, or connectivity). Usage nodes are used to provide a consistent naming over different electrical systems and different development branches.

3.4.2 Supporting the selection of terminals and related parts

In many companies, the selection process of concrete terminals, cavity seals and cavity plugs is done automatically on the basis of an abstract contacting specification and a components library. Against this background, the VEC data format specification must enable the description of all data those component libraries typically contain. At least, the following information must be able to be described:

- Existing compatibility between cavity and terminal

- Existing compatibility between cavity and cavity plug
- Existing compatibility between cavity and wire (type)
- Existing compatibility between terminal and wire (regarding core and insulation)
- Classification of a terminal as multi crimp able
- Material information of terminals (basic material, plating material of terminal reception and wire reception)
- Compatibility of a cavity seal in dependency of cavity, terminal and wire
- The suited contacting method (crimp, weld, …)

Note: The VEC data format specification must enable compatibility relationships to be explicitly defined (alternatively by instances-based or grouping-type-based compatibility statements) as well as implicitly by type specific (cavity, terminal, …) attributes and its values.

3.4.3 Design relevant attributes

In many companies, the part descriptions within component libraries contain some technical detail information that are relevant for a designer when he has to search and chose a dedicated part with dedicated technical properties. Against this background, the VEC data format specification must enable the description of the most common technical properties that are typically needed for this purpose.

Below, a few examples of such properties are listed. The list contains only properties in addition to the chapters 3.1, 3.3.1 and 3.4.1.

- All parts concerning
    - The robustness against water, oil, petrol, …
    - The permitted temperature range
- In case of wires respectively cores
    - The minimum bend radius
    - The permitted voltage range
    - The insulation material and thickness
- In case of terminals
    - The gender
    - The pull-out force
    - The permitted current range information in dependency of nominal voltage and core cross section area

3.4.4 Parts classification

The VEC data format specification must have a dedicated parts classification concept which allows a parts type specific description of technical properties. Unlike the KBL the VEC data format specification shall allow parts to be member of more than just one parts classification.

Afterwards, some examples of parts are listed that must be considered by the VEC parts classification concept.

- Connector housings: female / male and mixed housings, pin connector sockets

- Terminals: female and male pluggable terminals, ring terminals, battery terminals, coaxial terminals, contact bridges, bifurcated contacts, optical fibre contact, end-to-end connectors (core separating), Insulation displacement connectors (non-core separating), comb connector
- Wires: single core wires, multi core wires (regarding possible twisting and shielding)
- Wire protections: tapes, sleeves and tubes, fittings
- Seals: cavity seals, cavity plugs, and sealing mats
- Fixings: clips and cable ties
- Grommets: bend protection sleeves, rubber grommets
- E/E-Components: ECUs, sensors, relays, antennas, batteries, fuses, and component boxes

3.4.5 Description of OEM-parts interrelated with its components supplier(s) and manufacturer(s)

The VEC data format specification must allow the description of OEM parts interrelated with its components supplier(s) and manufacturer(s). For this, it must be possible to describe a mapping of the relevant part numbers. However, for some special cases, the VEC data format specification must allow the definition of mappings that consider further details as well.

3.5 Additional requirements out of the geometry/topology perspective

3.5.1 Topology

The VEC data format specification must have a concept to describe topology. This concept must consider

- Topology nodes (without coordinates)
- Directed topology segments which are related to the topology nodes
- Further data like segment length and segment cross section area

The topology concept must be designed as the interrelating element between 2D and 3D views.

The topology concept must enable a stand-alone description of topology information as well as a description in combination with relating part placement information and geometry views (2D and/or 3D).

3.5.2 Partitioning and mapping of topology

The VEC data format specification must have a partitioning respectively grouping concept for topology.

Furthermore, the topology concept must have a mechanism to describe mappings respectively topology embeddings. This can be relevant e.g. for a part with an own topology specification being integrated into the topology of a wiring harness.

A harness definition including topology might be used twice in the vehicle (e.g. left and right rear door). The topology concept must have a mechanism to express that the topology of both wiring harness instances obeys the same product definition.

3.5.3 Placement

The VEC data format specification must have a placement concept which allows the description where part instances or part usages (a kind of placeholder for a concrete part instance respectively a set of concrete part instances representing configuration dependent alternatives) are located on the topology. The placement concept must consider placements that are either defined by one or more reference points (e.g. for the placement of a cable channel) or a dedicated route (e.g. for the placement definition of a winding). In case of overlaps the required layering must be able to be specified.

3.5.4 Hierarchical Structuring

The VEC data format specification must have a concept which allows the definition of hierarchically structured segments. There are areas of application where the inner structure of a segment matters (e.g. some wires are taped together with another group of wires that are placed in a tube and around all of them a fixing is placed).

3.5.5 Topology zones

The VEC data format specification must have a concept to assign topology nodes and segments to specific zones that can be associated with properties or requirements (e.g. hot or wet areas, crash zones). It is possible that segments are not completely in or out of zones.

3.5.6 Dimensioning and tolerance specification

The VEC data format specification must have a concept which allows explicit dimension definitions between in each case two dedicated locations on the topology (specifying the length of the centre line). For this, it must be possible to address the following elements as dimension anchor points:

- Topology nodes
- The reference points that are used for the placement specification of part instance or a part usage. This includes the start and end points of routes (e.g. taping routes) as well.

In case of dimensions between two anchor points which allow ambiguous route interpretations (because of two or more possible paths) the required path must be able to be specified.

Each dimension definition must offer the possibility to specify a nominal value as well as an upper and lower boundary as tolerance specification.

3.5.7 General requirements concerning geometry views

The VEC data format specification must have a geometry view concept. The same topology must be able to be described in several geometric views. This includes

- 3D views (which normally show the target-installation of the wiring harness)
- 2D views (which normally present a dimensional drawing)

The geometry view concept must allow each topology relevant view element to unambiguously relate to its corresponding topology counterpart element.

Analogously for topology, the VEC geometry view concept must include a partitioning concept. This means the description of a geometry view must be able to be separated

in several partitions. At the same time, it must be possible for each of these partitions to be referred by several geometry views. An example use case for the latter could be one 2D drawing for a right-hand-drive vehicle and another 2D drawing for a left-hand-drive car referring (which means sharing) the same partition for the elements that belong to the rear.

3.5.8 3D view

The VEC geometry view concept must enable the description of 3D views of wiring harnesses without further external documents (e.g. 3D-PDF). The following information must be able to be described:

- 3D geometry nodes together with their 3D-coordinates (e.g. the coordinates addressing the target-installation of the wiring harness)
- 3D geometry segments each referring their start- and end 3D geometry node, the tangent vectors at these nodes and the definition of the centre line (BSpline). Further details (segment length and cross section area) must be able to be unambiguously assigned in combination with a related topology description.
- Position and orientation of part instances or part usages (connector housings, fixings, …). Note: As the required placement concept as specified in chapter 3.5.3 is demanded to be based on the coordinates-less topology information it can only describe axial information which means placements in relation to the centre line. This means the placement concept will be neither capable to express radial orientation information of placed parts nor translations between bundle position point and the origin of the placed element.

3.5.9 2D view

The VEC geometry view concept must enable the description of basic 2D views of wiring harnesses without further external documents (e.g. SVG). The following information must be able to be described:

- 2D geometry nodes together with their 2D-coordinates and optionally a sheet reference
- 2D geometry segments each referring their start- and end 2D geometry node and giving a rough definition of the connecting line
- Position and orientation of part instances or part usages representing view items

3.5.10 Extra requirements addressing variance information

Out of the topology/geometry perspective, at least the following elements must be able to express variance dependency by referring to a certain variant configuration term

- Topology and geometry nodes and segments
- Part instances and part usages
- Placement and dimension specifications
- In case of a geometry view divided into partitions each of those partitions

3.6 Additional requirements out of the connectivity perspective

3.6.1 Abstraction layers

The VEC data format specification must have a concept for the following three abstraction layers

- Architecture
- Schematics
- Wiring

Concerning the data related to one of these abstraction layers, the concept must enable the description both stand-alone and together with the existing interdependencies.

3.6.2 Architecture

The following information must be able to be described:

- Network nodes: representatives for actors in the electric system, e.g. actuators, sensors and ECUs
- Network ports: representatives for the pins of a network node
- Nets: representatives of connectivity between the related network ports which means without further electric consumer or transformer in between. Nets shall not make any assumptions about the physical realization of the connection and about the topology. On the basis of nets, it must be possible to define net groups.

3.6.3 Schematics

The following information must be able to be described:

- Component nodes: representatives for elements in the electric system, e.g. actuators, sensors, ECUs and in comparison, with network nodes additionally inliners and splices. Furthermore, there must be a concept for the rough description of the internal architecture of a component node.
- Component ports: representatives for the pins of the component node
- Connections: representative of a connectivity between the related component ports out of the perspective of a single core wire connection. Connections shall not make any assumptions about the topological representation. On the basis of connections, it must be possible to define connection groups and routing definitions (see chapter 3.3.7).

3.6.4 Wiring

With regard to the development process, the wiring is the immediate predecessor of the wiring harness definition. With regard to content, the wiring does normally not yet consider geometry aspects.

Behind this background, the VEC wiring concept must enable the definition of all necessary requirements that enable together with a corresponding geometry definition the determination of the concrete part instances (in terms of the KBL “occurrences”). Afterwards, some examples are listed the VEC wiring concept must consider at least. However, which pieces of information are regarded as necessary requirements in

detail are normally process specific and the VEC wiring concept must be flexible enough to cope with that.

- EE components: e.g. existing housing and pin components
- Connector housings: e.g. number of slots and cavities
- Terminals: e.g. special pin properties, definition of co-axial terminals
- Wires: e.g. core cross section area, insulation thickness, colour
- Contacting: e.g. the type of contacting, see chapter 3.6.5
- Cavity mounting: the assignment of terminals to cavities, see chapter 3.6.6
- Mating: the relationship between pin and contrary pin, see chapter 3.6.7

3.6.5 Contacting

The VEC data format specification must have a concept that considers the following types of contacting

- Simple contact: assignment of a terminal (wire reception) to one wire end, in case of pluggable terminals mostly in combination with a cavity mounting definition (see chapter 3.6.6)
- Multi contact: assignment of a terminal (wire reception) to more than one wire ends, in case of pluggable terminals mostly in combination with a cavity mounting definition (see chapter 3.6.6)
- Open wire end: definition of no contacting
- Inliner: a special disconnection point between two or more wiring harnesses.
- IDC (insulation displacement connector): connection between two or more EE components that is realised by one physical wire
- Splice: a connection of two or more wire ends without connector housing or EE component

3.6.6 Cavity Mounting

The VEC data format specification must have a cavity mounting concept in order to describe

- Simple mountings: assignment of a terminal (terminal reception) to one cavity
- Contact bridge: Assignment of a terminal (terminal reception) to more than one cavity
- Plug-in bridge: Assignment of a terminal (terminal reception) to more than one cavity that is itself not connected to a wire.
- Line bridge: Two terminals that are each assigned to one cavity and at the same time are interconnected by a short wire.
- No cavity mounting: e.g. in case of a pluggable terminal that is directly connected with its counterpart terminal (see chapter 3.6.7).

3.6.7 Mating

The VEC data format specification must have a mating concept in order to describe the (normally pluggable) connectivity between

- The pins/terminals (in case of coax-terminals and piggyback terminals even the relevant terminal receptions) of an EE component and the contrary pins/terminals of the harness connector
- The pins/terminals (in case of a coax-terminal even the relevant terminal receptions) within the two connector housings of an inliner
- Bolts and ring terminals

3.6.8 Extra requirements addressing variance information

Out of the connectivity perspective, at least the following elements must be able to express variance dependency by referring to a certain variant configuration term

- All elements of the abstraction layer architecture
- All elements of the abstraction layer schematics
- All elements of the abstraction layer wiring including usages of EE components, connector housings, terminals and wires as well as contacting-, cavity mounting- and mating-definitions.

3.6.9 Pinning Information

The VEC must provide the possibility to describe electrical interface of EE components. This includes:

- Signals supported by a pin
- Voltage and current values of a Pin for different types and times.
- Variant dependant behaviour of the pin due to the software deployed on an ECU.

3.7 Cross-sectional requirements

3.7.1 Process specific information

The VEC data format specification must have a concept which describes how to enrich a VEC compliant file with process specific data. For this, each VEC element is requested to be capable to be extended with user-defined attributes.

3.7.2 Multi-language support for descriptions

For attributes, that carry non-identifying informal text (normally descriptions), the VEC data format specification must have a concept for multi-language support. The language code must be compliant to ISO-639.

3.7.3 Free choice of measurement units for dimensional quantities

For attributes, that carry dimensional quantities, the VEC data format specification must permit free choice of measurement units. In order to ease the data import, the VEC data format specification must have a special concept for SI data units (like metre – m) and scaled units (like millimetre – mm) as well as combined units (like kg per metre – kg/m).

3.7.4 Orthogonal grouping concept

The VEC supports information objects from different domains within the development process (e.g. system schematic, component data, wiring harness data, geometry). To

support general traceability use cases the VEC shall provide a concept to assign information objects from different domains to groups (e.g. all elements that relate to a specific customer function, a safety relevant requirement).

This concept shall be independent (orthogonal to) of the domain-oriented structure of the model.

3.7.5 Application Constraints

The VEC data format specification must have concept to apply constraints on the application / usage of specific elements defined in the VEC. This concept must be based on instances (e.g. component nodes in a system schematic or occurrences of a connector in a wiring harness). It is required to define the scope of validity of a certain design / construction and to be complementary to the requirement mentioned in chapter 3.1.2. However, the information required to define the constraints are the same.

3.8 Other / technical requirements

3.8.1 Data format

The VEC data format is required to be an XML based format. Behind this background, the VEC data format specification must include an XML schema definition. The VEC.XSD must define UNICODE-coding in order to enable special characters to be exchanged.

3.8.2 Binding the VEC with external models and formats

There are many use cases in which the information represented by a VEC is related to elements outside of the VEC. This relationship to external elements is in many cases relevant information for downstream participants in the process and should be preserved. One obvious approach to achieve this would be to integrate this information into the VEC Model. However, this approach is not satisfying in many cases, because the respective information might not be within the central scope of the VEC or other established standards exist for the representation of this kind of information. Examples for such information are:

- Visual representations like:
    - 3D-Models
    - Wiring Diagrams
    - Component-Symbols
    - Drawings
- Requirements

In order to be able to use these external formats together with the VEC a solution is necessary to create a link / mapping between information in the VEC and information in other external formats.

To make this requirement more clearly, it is explained in the following with a small example:

The Part Master Data of a connector can be described in VEC in a structured way (Cavities and their technical properties, segment connection points, etc.). However, there are many use cases where a visual representation of the connector is needed (e.g. Harness Drawing, Wiring Diagram). This visual representation is often distributed as an SVG-Symbol. If there is no link between the information in the VEC and the visual representation, the information can only be used in a limited way. For more advanced use cases it is necessary to know which element in the VEC (e.g. a Cavity) is represented by which graphical element in the SVG (e.g. a square).

In detail the following requirements must be satisfied:

1. It must be possible to create a mapping between any kind of information in the VEC and elements in an external data source.
2. If the VEC is mapped with some external information, then it must be still possible to read both formats independently (without using the link). E.g. if VEC is linked with an SVG, it must be still possible to read the VEC, without interpreting the SVG and it must be still possible to read SVG with an appropriate viewer, without knowing about the VEC.
3. The used mapping mechanism must not require any structural information about the described data outside of the VEC. Master and source of this kind of data is the VEC. This means for example if a VEC should be linked to SVG it must not be necessary that the SVG complies with a defined structure and has to use certain elements for the representation of its information.
4. The mapping mechanism must be usable for different types of formats. However not all formats are suitable to be used for a mapping. At least they must be structured in some way and elements must be identifiable (e.g. JPEG is not appropriate for such mapping mechanism).
5. The mapping must be standardized, in order to allow an exchange of the mapping information between different systems.

3.8.3 Tracking of changes to the digital representation of documents

The VEC data format specification must provide a concept to detect / track changes to digital representation of a document. There are cases where the digital content of a document within the VEC changes, even if the PDM relevant information remains unchanged (e.g. the document itself has not been republished, but changes to the exporting system resulted in content changes).

3.8.4 Extendable Enumerations

The data model of the VEC must support extendable and fixed enumerations (open & closed). For closed enumerations all possible literals are known at the time of the definition of the VEC. For open enumerations some literals are known (which should be standardized), but new literals might emerge in the future. This is often the case,

when the literals relate to technical aspects. With new innovations in technology, new literals might be required and shall not require a new version of the VEC schema.

In detail the following requirements must be satisfied:

1. Closed lists of enumeration values have to be supported.
2. Open lists of enumeration values have to be supported.
3. The VEC UML model shall be the single source for documenting open and closed enumerations.
4. The concept shall be supported in the XML schema, too.
5. It shall be possible to validate closed list values against the XML schema
6. It shall be possible to validate the pre-defined open list values against a specific XML schema

4 General Guidelines

The most of definition of the VEC is contained in chapter 5 as a detailed model description. However, there are general concepts and guidelines that apply universally and are not limited to an individual model element. Therefore, they are not formulated as part of the model description. These guidelines are defined in the below sections and shall be followed for all implementations of the VEC.

4.1 Handling of Identifiers

The VEC and its XML Schema offer different concepts for the identification of model elements addressing certain requirements and those shall be used accordingly.

4.1.1 Id Attributes

All *xs:complexType* define an id-Attribute with the type *xs:ID*. These are technical ids that are necessary for the referencing mechanism of the VEC within a single XML file. The semantics, constraints and requirements are defined by the XML Standard and XML Schema itself. These ids do not have any significance outside a VEC file.

4.1.2 Identification-Elements

Many types defined by the VEC have an “Identification” sub element (E.g. the *PartOccurrence*). This is meant to be a semantic identifier of the object represented by the VEC element. The following rules apply to those identifiers:

1. The expectations defined in the documentation of the VEC model of the corresponding attribute shall be ensured.
2. The identifications shall be unique for a certain element type, at least within its context element. In other words, the VEC model and its representation as XML Schema is a hierarchical data model. That means, that an identification shall be at least unique within its direct parent element (e.g. the identification of a *HousingComponent* shall be unique within its *EEComponentSpecification.*
3. Two elements of different types can have the same *Identification*. However, this is only recommended, when the two VEC elements represent the same domain entity from different points of view, otherwise this shall be avoided as far as possible.
4. In general, it is recommended to keep the *Identifications* stable over the time. This means, that if an object is exported multiple times the *Identification* of it should be the same. However, this is not possible in all cases, for all processes and all tools. Therefore, a process and / or tool creating VEC files should describe for all elements, under which conditions *Identifications* are stable or new ones are created.

4.1.3 AliasIdentifications

Certain elements have the possibility to define AliasIdentifications in addition to their unique identifications. These are identifiers of the object in a different scope, system or process. One use case of this kind of ids is the creation of traceability links.

Examples for usages of the AliasIdentification are:

- The identifier of a connector in the electrological process (with geometric variants)
- The identifier of a node or segment in a MCAD tool
- An assigned UUID of an element.

4.2 Extension Mechanisms

If the well-defined data structures and fields are not sufficient for the specific needs of a process or a tool, the VEC provides powerful extension mechanisms. Namely the extension mechanisms are custom properties and open enumerations (see the corresponding chapters in the model description).

However, it should be considered that information transported via these mechanisms is not standardized and is always subject to an individual agreement between interface partners. Therefore, these mechanisms shall be used with extreme caution.

It is strictly forbidden to use these mechanisms for the transfer of information that is already standardized within the VEC. In particular it is not permitted:

- To store information in custom properties where already well-defined concepts exist in the VEC to store the same information, e.g. using a custom property instead of an attribute or a more specific class in inheritance tree.
- To use self-defined OpenEnumeration-literals when well-defined literals with the same semantics already exist.

VEC-Files that do not obey to these rules are noncompliant to this data format specification.

If the extension mechanisms are used, it shall always be considered if these extensions might be a valid feature request for the VEC Standard.

4.3 Type Inheritance

The VEC uses an object-oriented class and inheritance concept. The following clarifications apply to its use:

- Only non-abstract classes can be instantiated.
- In an inheritance hierarchy, the choice of the used type represents a semantic information itself. For example, the usage of a PluggableTerminalSpecification is a more specific information than the usage of a TerminalSpecification. It is not required to use the more specific class if the information is not available or it should not be transmitted. However, it is not permitted to use the more general class and transfer the information of the more specific class in a custom property, or similar (e.g. use the TerminalSpecification with a custom property “type=pluggable”).

4.4 Default- and Missing-Value Handling

For various reasons, there may be attributes of entities where no value can be exported, or a special semantics is required. The cases are:

- The information is not supported by the system / process. So, it is never available for this system / process.
- The information is supported by the system; however, the value is not defined by the user.
- The information is explicitly defined as “arbitrary” for the use case (e.g. the part version in a bill of material or a compatibility statement).

All cases might exist for mandatory attributes as well as for optional attributes. Due to the design, numerical values in the VEC and its high level of optionality the following definition of special values should be only relevant for *string*-Attributes:

	Mandatory Attribute	Optional Attribute
Unsupported	<tag>/NULL</tag>	omitted tag
Undefined	<tag></tag>	<tag></tag>
Arbitrary	<tag>/ANY</tag>	<tag>/ANY</tag>

- **“/NULL” & “/ANY”** means, that the attributes with the name “tag” in the VEC receive these values.
- **<tag></tag>** means, that an attribute with the name “tag” and an undefined value is represented in the VEC as an existing XML element with no value (no contained text() node).
- **Omitted tag**: means the element tag for the attribute is not present in the VEC.

4.5 Instantiation of Model Structures

There are various locations in the VEC model where structures / patterns are defined and used / instantiated somewhere else (e.g. a connector with its slots and cavities). In most cases, the elements in the definition of a structure have corresponding elements in the instancing (e.g. ConnectorHousingSpecification → ConnectorHousingRole, Slot → SlotReference & Cavity → CavityReference).

In cases where defined structures are instantiated, these structures shall be instantiated completely. That means, for every element in the structural definition a corresponding element in the instancing shall exist, regardless if it is used in the respective VEC or not (e.g. for each Cavity of a ConnectorHousingSpecification, a CavityReference in the corresponding ConnectorHousingRole shall exist). This applies to the following list of structures, which is here for reasons of clarification and which is not exhaustive:

- Connectors
- Wires
- EEComponents
- CompositeParts (e.g. Assemblies or Modules)

4.6 VEC-Package

4.6.1 Background

A Vehicle Electric Container (VEC) is a single XML file following the structure defined in the VEC XML schema. It contains all the information of a harness, a set of harnesses, or other related information defined in the VEC specification. A VEC Container can reference other files via the DocumentVersion element and information contained in other files via the ExternalMapping concept.

There are use cases where one wants to exchange the VEC together with these referenced files. There is also the need to exchange a set of VEC files together. The VEC-Package addresses these use cases and specifies the mechanism to exchange VEC files and any associated files as a package.

4.6.2 Detailed Solution

A VEC-Package is an archive containing at least 2 files:

- One index file (a VEC file)
- One data file (not required to be a VEC file)

Depending on the individual requirements the technical format of the archive can be:

- TAR
- ZIP
- or a zipped tar.

In addition, the archive can contain any number of further data files. There are no restrictions on the type or format of these files. A VEC-Package may contain further VEC files. Or it may contain drawings as SVG, CAD models of the harness or of connectors as JT models, for example.

The structure of the archive is not restricted. A VEC-Package may contain a flat set of files but may also have a folder structure. It is recommended to use a folder structure to organize the files in the archive: e.g. grouping of all drawings, project specific groupings.

There is no naming convention for files and folders inside the VEC-Package defined. It is up to the user to name a folder or a file. However, it is recommended to use the known and established file name extensions for the files in the package. I.e., “.vec” for a VEC file, “.svg” for a SVG file, or “.jt” for a JT file.

A VEC-Package shall contain an index file providing further information about the context of the package. The index file has the reserved name “index.vec”. As the file name suffix already suggests, the index file is a valid VEC file, conforming to the VEC XML schema.

The elements of the index VEC file are restricted to the classes DocumentVersion and PartVersion. The index file contains a DocumentVersion for each file in the package.

The attributes of the DocumentVersion are used to provide further information on the files:

- dataFormat: the format of the file in the VEC-Package (as MIME-Type if available).
- documentNumber: the number of the document
- documentVersion: the version of the document
- fileName: the name of the file as it appears in the package, including the folder structure

A DocumentVersion may reference one or more PartVersion objects via *referencedPart* to give further details on the usage of the file. For example, the fact, that an SVG file which represents the wiring diagram of a harness, can be expressed in the index file by a DocumentVersion pointing to a PartVersion, which represents the harness.

5 Model of the VEC data format

5.1 Key Concepts

5.1.1 Parts and Documents

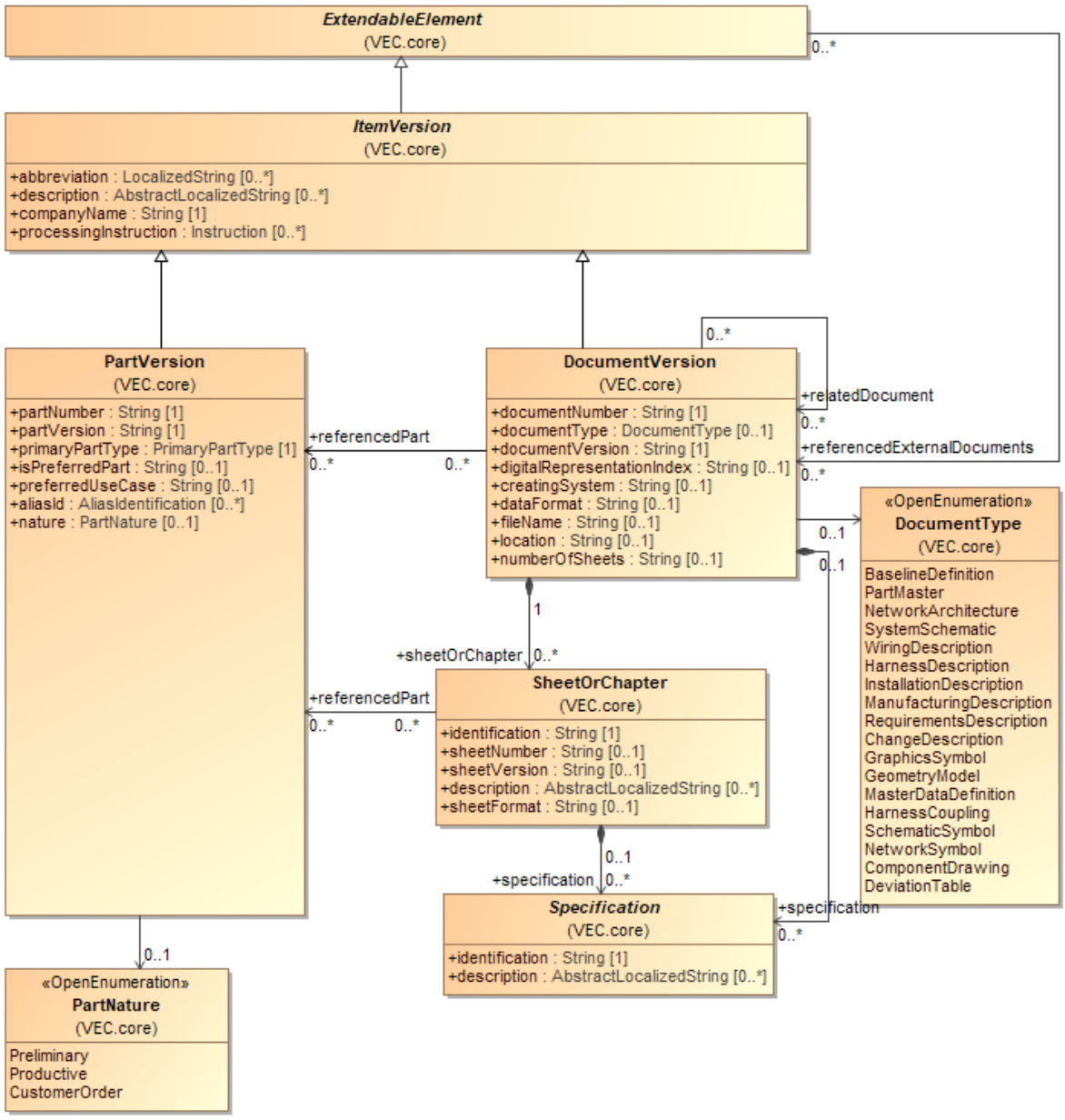


Figure 1: Parts and Documents

DocumentVersion and *PartVersion* are representing one of the most fundamental key concepts of the VEC. They are the only classes in the VEC model that are defining version information. In combination with the attributes *companyName* and *partNumber* or respectively *documentNumber* they define an unambiguous context for data exchange. If two instances of the VEC are containing *PartVersions* or *DocumentVersions* with equal values of the attributes *companyName, partNumber* and *partVersion* or *companyName, documentNumber* and *documentVersion,* then the two objects can be trusted to be unchanged.

Unchanged in this context does not mean binary identical content of the resulting XML-Subtree, but unchanged user data. So if the content of a *DocumentVersion* is transferred from one VEC-file to another (e.g. for documents describing part master data), the binary representation can be different (e.g. order of the xml elements, technical ids for xml idrefs), but the information represented by the *DocumentVersion* must not be different (e.g. property values of a *Specification*). However, it is explicitly allowed that the scope of the information contained in different instances of the same *DocumentVersion* may differ. For example, in most processes a component (e.g. a connector) is specified by a technical document, in the VEC represented as a *DocumentVersion*. In the use case of part master data exchange the VEC for the component might contain very detailed information. For the use case of harness description not all of this detailed information is required (e.g. for a connector the information about slots and cavities might sufficient) and therefore it should not be included. However, the component is still specified by the same technical document. Another example of a scenario where this might occur, is the update of a VEC exporting system. A later export might contain more information, even if the original document has not changed.

These cases are allowed with an unchanged *documentNumber, documentVersion,* if the shared part of the represented information in both *DocumentVersion* instances is still consistent.

More specific possibilities to track changes of *DocumentVersions* are supported by the *digitalRepresentationIndex*. In some cases, the *DocumentVersion* represents a version of a real document in the domain. The *Specifications* in the *DocumentVersion* represent the values defined in the original document. Thus, the *DocumentVersion* in the VEC is not a managed document by itself, but only a specific digital representation of another document. In other words, it is only a translation of the original document into the language of the VEC. For example, there is a connector described by a drawing / data sheet, which is available in some arbitrary format (e.g. PDF). This document has a number and a version. When the document is transferred (e.g. with a converter or manual input) into the VEC, the *DocumentVersion* containing the *Specifications* might have the same *documentNumber* and *documentVersion* as the original data sheet, since it is not a managed document in the process, but only a different representation of the same information. In such a scenario, it is possible that the content of a *DocumentVersion* changes without a change of the original document (e.g. there was an input error during the manual transfer of information, there is a new version of the converter / exporter and many more). However, the original document has an unmodified *DocumentNumber & DocumentVersion.* To indicate such (potential) changes of the content of a *DocumentVersion* the *DigitalRepresentationIndex* was introduced. *DocumentVersions* that contain a different amount of information (see above) shall also have different *digitalRepresentationIndex* values (if defined).

That means, if no *DigitalRepresentationIndex* is defined, the rules from above apply. If a *DigitalRepresentationIndex* is defined, the content of the *DocumentVersion* can only

be trusted to be unchanged, if *companyName, documentNumber, documentVersion* and *digitalRepresentationIndex* are the same.

Since both classes can define PDM information they are derived from the abstract class *ItemVersion*.

The differentiation between *PartVersion* and *DocumentVersion* in the VEC is necessary, because a part can be described by multiple documents (e.g. a technical data sheet, a drawing and a 3D-model) and one document can describe multiple parts (e.g. a drawing for a contact family that displays multiple terminals and their corresponding seals). As a result a part can, be changed (requiring a new *PartVersion*) without the need of changing all of its describing *DocumentVersions.* For example, if a single property of a component changes, a new version of the part itself and its technical data sheet is necessary, but its 3D model might remain unchanged. Another example is, that a document describing multiple parts might be changed (requiring a new *DocumentVersion*), but in fact only one of its represented parts is changed (requiring a new *PartVersion*). This relationship between a *PartVersion* and its describing *DocumentVersions* is expressed in the model by the association *DocumentVersion.referencedPart*.

The relationship *DocumentVersion.relatedDocument* can be used to describe general relationships between *DocumentVersions* (e.g. the relationship between a technical drawing and a given standard the drawing is compliant with). The composition of *SheetOrChapter* considers the fact that *DocumentVersions* may be composed of several sheets or chapters.

The VEC is designed as a format for data exchange. This requires that all content data can be related to a certain unambiguous version context, in order to allow a receiver of the information to easily check if and what has changed since the last data exchange. For this reason, all content data that is not constant is defined with information specific subclasses of the class *Specification*. A *Specification* is always contained in a *DocumentVersion* and thereby related to an unambiguous version context.

5.1.2 Assignment Group

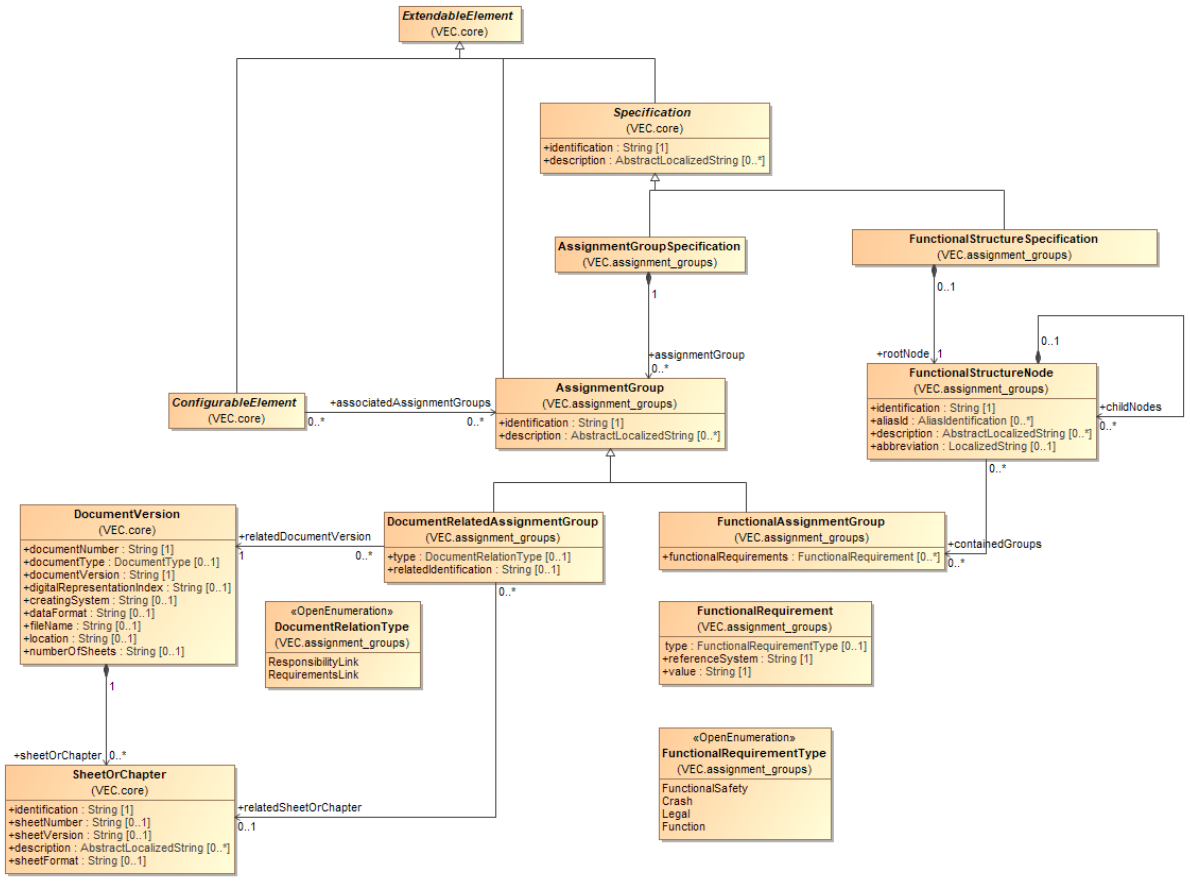


Figure 2: Assignment Group

An *AssignmentGroup* is a concept that allows the clustering of arbitrary elements in ways that are orthogonal to hierarchical and semantic structure of the VEC. Specific use cases are represented as subclasses of the *AssignmentGroup* and enriched with additional information. Proprietary groupings can be created with the AssignmentGroup itself and additional information in custom properties.

The *FunctionalAssignmentGroup* can be used to assign elements to a function across the different layers of abstraction.

The DocumentRelatedAssignmentGroup can used to relate elements to a specific document or an element within the document (e.g. a requirement).

The assignment to a group originates from the *ConfigurableElement* itself. This is designed intentionally and has two reasons:

1. In the use cases for this grouping concept it is intended, that the groups exist first (e.g. a function) and the assignment is created together with elements along the process. For example, a connection in a system schematic is assigned to a function. When a wire is derived from the connection, the wire inherits this assignment. Therefore, it wouldn't be practicable to be enforced to modify the *DocumentVersion* containing the *AssignmentGroup* for every added element.
2. In today's processes the assignment of elements to such groups are done in the same documents where the elements themselves are defined.

This concept adds another layer of extensibility to the VEC. *CustomProperties* allow the addition of user-defined properties to a single object. The assignment groups allow the creation of custom grouping of objects. However, it is not permitted to use this concept to express relationships that have well defined concepts in the VEC (e.g. assignment of occurrences to composite parts or usage nodes).

5.1.3 Variants

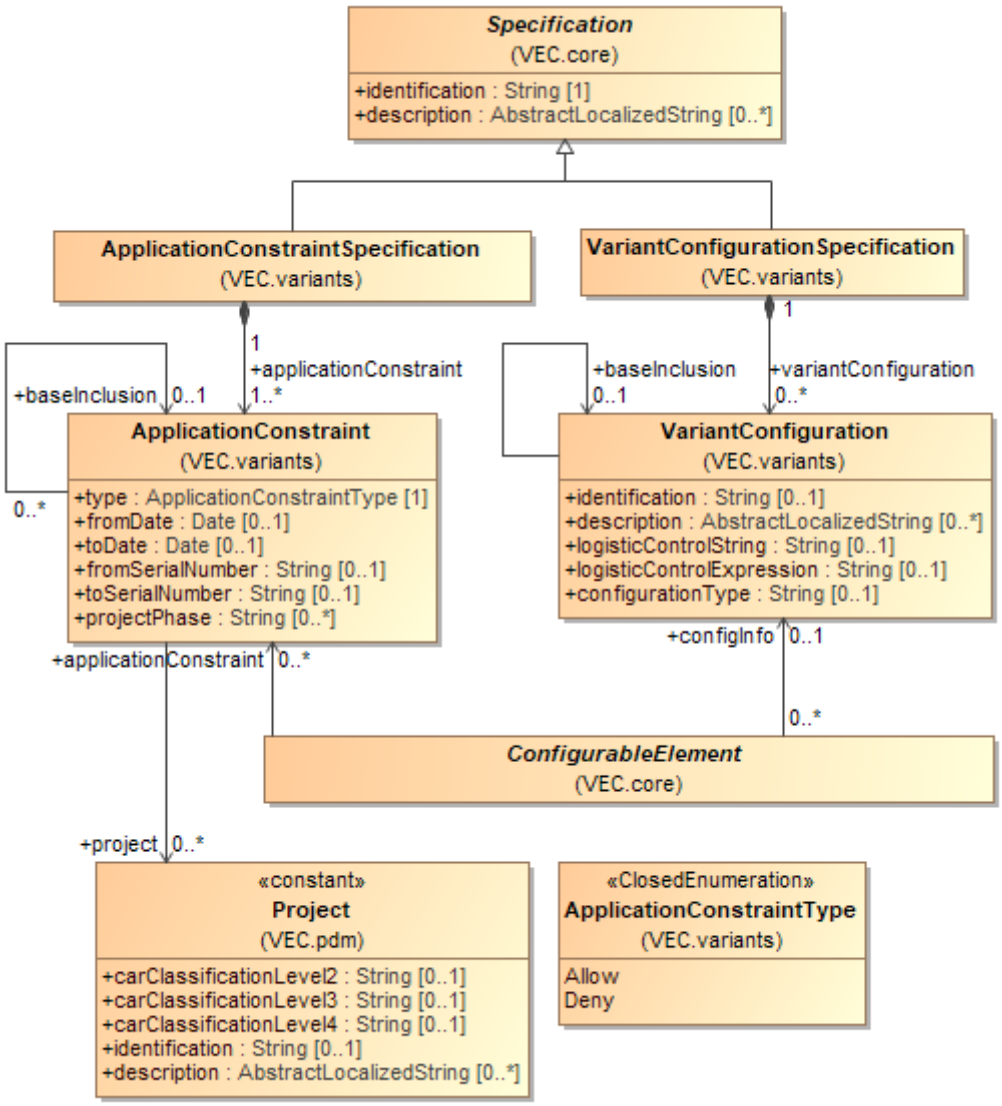


Figure 3: Variants

A *VariantConfigurationSpecification* is a container for various *VariantConfigurations*. Each *VariantConfiguration* describes a certain constraint which makes it possible to determine whether a relating *ConfigurableElement* is relevant regarding a certain configuration or not. All classes in the VEC that can reference information about their *VariantConfiguration* inherit directly or indirectly from *ConfigurableElement*.

The *ApplicationConstraint* is quite like the *VariantConfiguration.* It restricts or allows the application of a specific configurable element (e.g. a *PartOccurrence* or an element of a system schematic). It is complementary to the *VariantConfiguration* which is

defined in some form of control string and restricts the existence of an element based on the feature space (within a platform). The *ApplicationConstraint* is focused on the dimensions of time and product hierarchy in a concise semantic way (attributes and relationships).

Note: Please refer to the detailed class description for information about which elements inherit from *ConfigurableElement.*

Note: *VariantConfiguration.logisticControlExpression* has been introduced as an alternative attribute for *VariantConfiguration.logisticControlString*. For the *logisticControlExpression* the prostep ivip project group “VES-WF” wants to introduce a defined syntax in the future.

5.1.4 Variant Structure

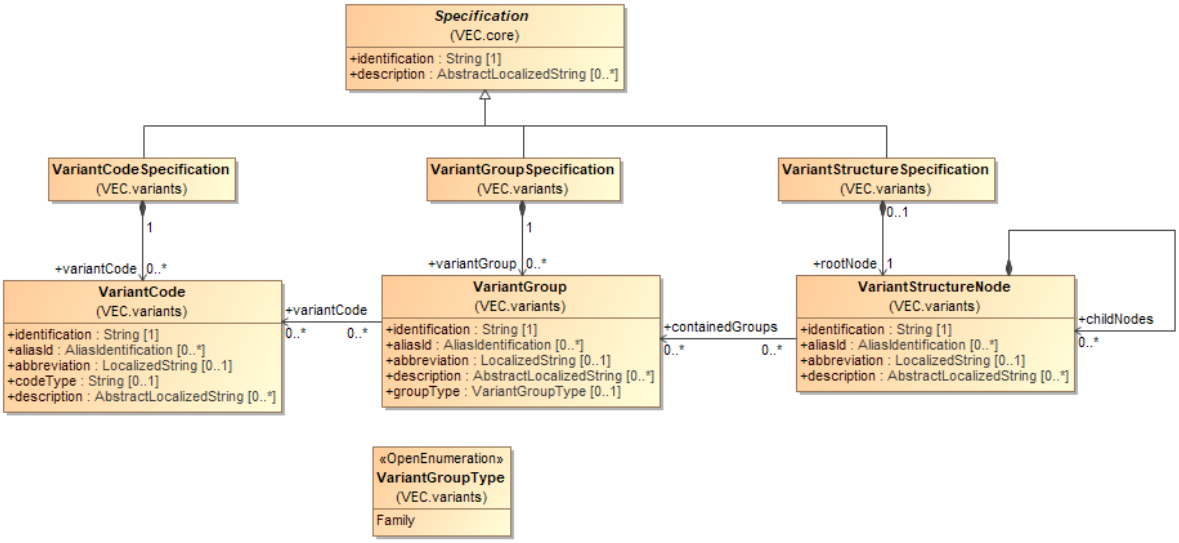


Figure 4: Variant Structure

The classes displayed in this diagram can be used to describe meta information about the elements used to described variance properties. The description of the variance information of a concrete element (*ConfigurableElement*) is done with a *VariantConfiguration* and the contained *logisticControlString*. These strings are normally created from *VariantCodes* and a proprietary boolean syntax.

A *VariantCodeSpecification* is a container for various *VariantCodes*. Each *VariantCode* describes an elementary literal which can be used as part of a *VariantConfiguration*.-*logisticControlString* or *VariantConfiguration.logisticControlExpression*.

A *VariantGroupSpecification* is a container for various *VariantGroups*. A *VariantGroup* describes a grouping for *VariantCodes*. There can be a wide variety of reasons to group VariantCodes. However, the most common is to mark *VariantCodes* that belong to the same family (mutually exclusive) variant codes.

A *VariantStructureSpecification* with its *VariantStructureSpecificationNode* can be used to define any kind of hierarchical structure upon the *VariantGroups*.

Note: The VEC does not constrain a *VariantCodeSpecification* or a *VariantGroup-Specification* to a precondition for use in *VariantConfigurations*. *VariantCodeSpecification* and *VariantGroupsSpecification* are intended as an optional means to exchange information independently about the “vocabulary” which is possible to express *VariantConfigurations*.

5.1.5 Usage Node

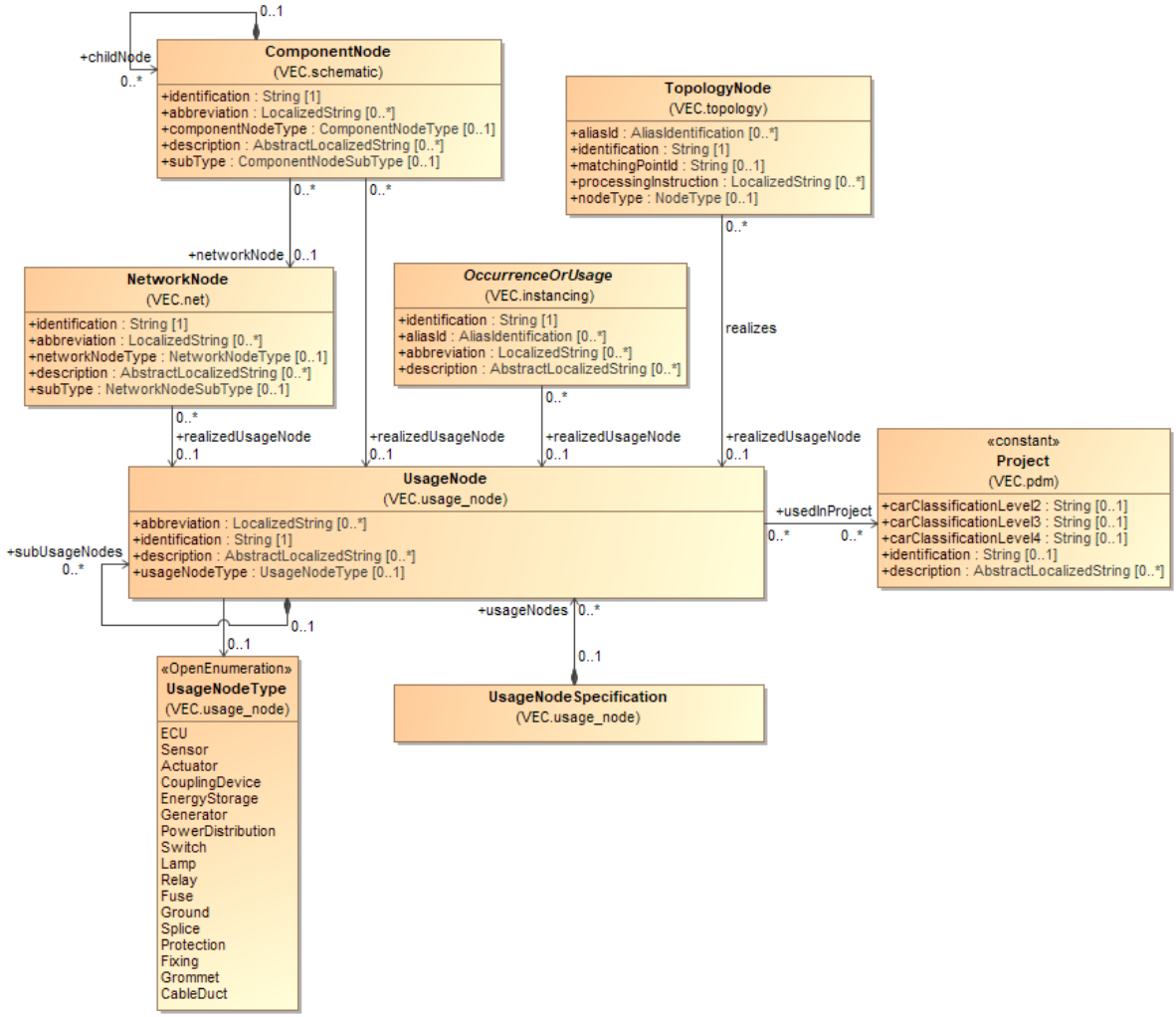


Figure 5: Usage Node

A *UsageNode* represents a position in an abstract vehicle, but it does not imply a certain location in space. For example, the "Head Light Left". *UsageNodes* belong to the master data and they are defined on some companywide level. They can be used to enforce consistent naming over different projects and different development streams (e.g. between Geometry and Electrologics).

A *UsageNode* can be realized by different elements in the VEC (e.g. *NetworkNode, OccurrenceOrUsage, TopologyNode, ComponentNode*).

5.1.6 Usage Constraints

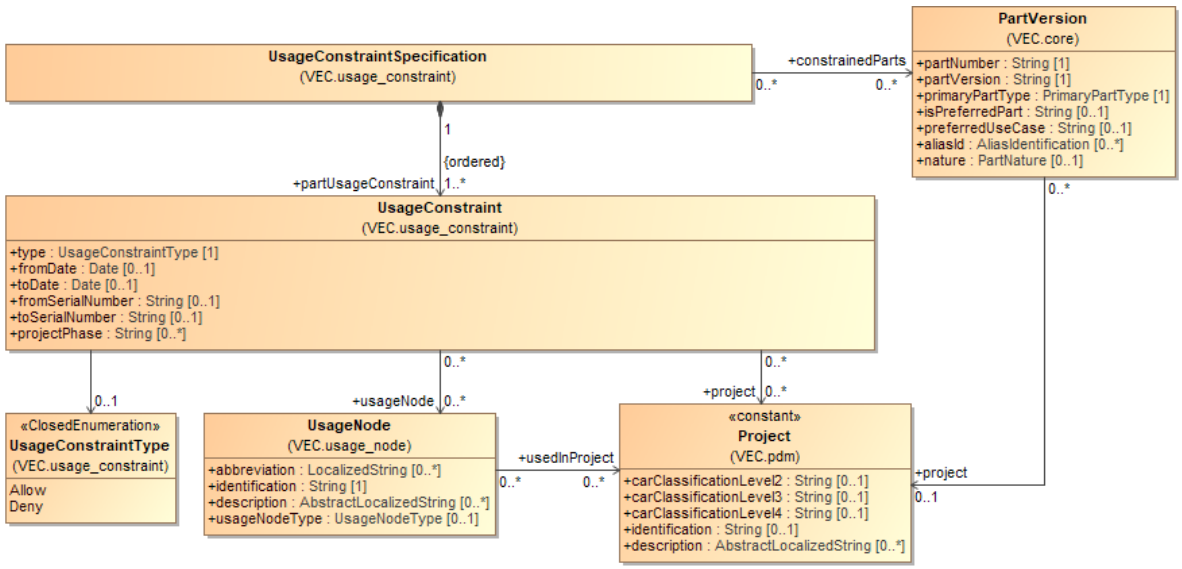


Figure 6: Usage Constraints

A *UsageConstraintSpecification* is intended to describe the allowed usage of the *constrainedParts*.

A *UsageConstraintSpecification* is a container for various *UsageConstraints* each representing a single constraint. The sequencing of the *UsageConstraints* specifies the priority: a general denial may be relativized by a following *UsageConstraint* and vice-versa. In other words, the ordering of the *UsageConstraintSpecification.-partUsageConstraints* collection has a semantic and is highly relevant. *UsageConstraints* further back in the collection have a higher priority than *UsageConstraints* further ahead. Successive *UsageConstraints* of the same type shall be interpreted as concatenation, whereas successive *UsageConstraints* of different types shall be interpreted as exceptions to the preceding statement.

The relationships U*sageConstraint.usageNode* and *UsageConstraint.project* make it possible to specify at which *UsageNodes* respectively in which *Projects* the usage of the constrained *PartVersions* is allowed/denied.

5.2 Basic Datatypes

The VEC supports the following a number of primitive standard data types:

- Boolean
- Date
- Double
- Integer
- String

In addition, the VEC defines various more complex general-purpose data types that are used within the specification. Those are defined in the following sections.

5.2.1 Localization of Strings

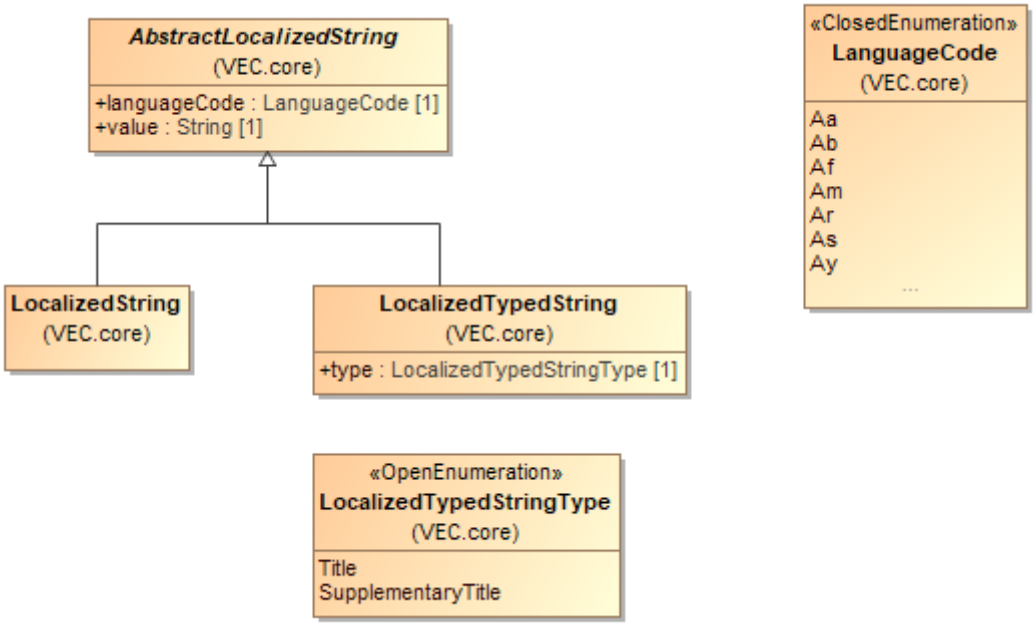


Figure 7: Localization of Strings

The VEC provides a concept for the localization of string values. Every attribute that can be localized has the type of *AbstractLocalizedString* or one of its subclasses.

An *AbstractLocalizedString* consists of an ISO *languageCode* and a *value*. Attributes of this type have a multiplicity of [0..*].

In the case of a *LocalizedString* the attribute represents a human readable text with a specific semantic. All *LocalizedString* instances for a single attribute must represent the same meaning in different languages.

When a *LocalizedTypedString* is used, the semantics of the attribute values can be further detailed by the *type* attribute. For example, an *ItemVersion* can have more than one description with a more specific semantic of each description type. In this case, all *LocalizedTypedString* instances for an attribute must represent the same meaning for their type value in different languages.

5.2.2 Numerical Values & Units

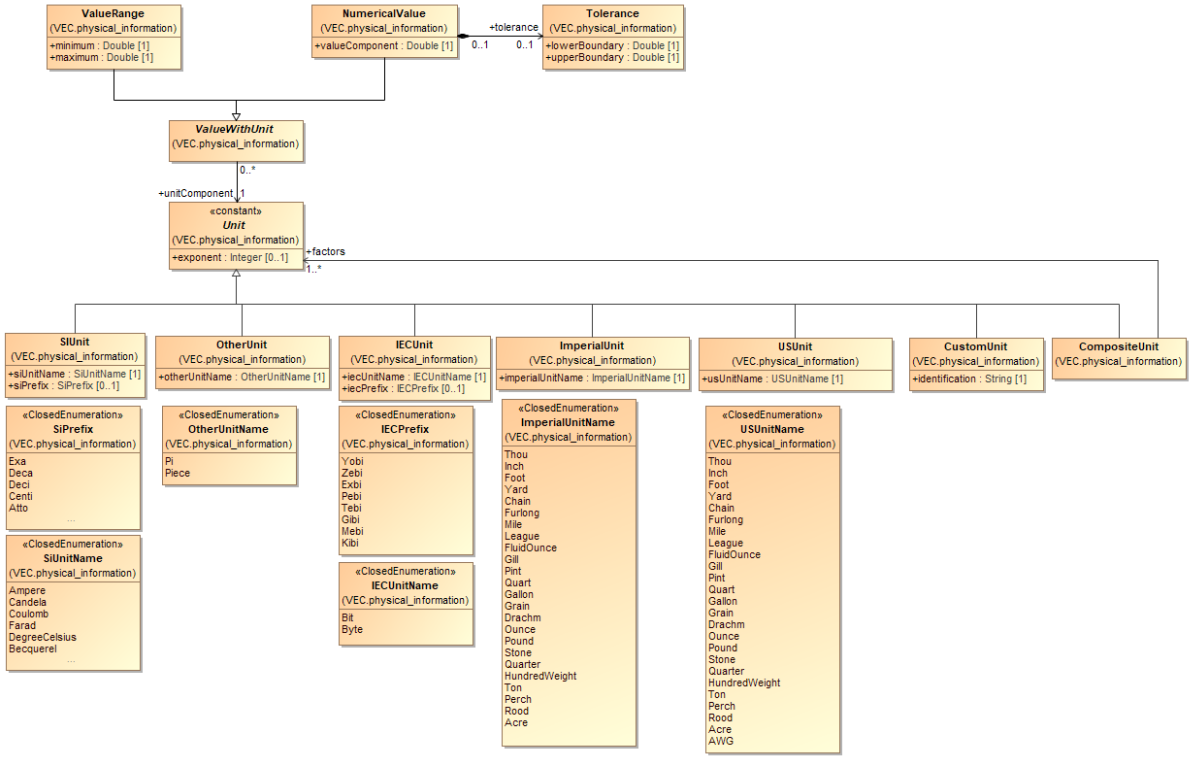


Figure 8: Numerical Values & Units

With Version 1.1.0 a more flexible and meaningful unit system has been introduced. A unit is used to define the dimension of a numerical value. In the VEC a unit is expressed by the abstract class *Unit*. The different existing types of units are represented by its concrete subclasses. Every unit can have an optional exponent. If no exponent is defined, this is interpreted as an exponent of 1.

The class *SIUnit* defines a unit in the terms of the SI-System. A *SIUnit* is defined by a combination of an optional *SiPrefix* (e.g. Milli) and *SIUnitName* (e.g. Metre).

The class *IECUnit* defines a unit in the terms of the IEC-System which is used for measurement of digital data amounts.

The class *ImperialUnit* defines a unit in the terms of the imperial unit system (e.g. Inch).

The class *OtherUnit* is used to define units that do not belong to any standardized unit system, but which are relevant in the context of the VEC. Currently these are the units Pi (as a unit for angular frequency or circular measure) and Piece.

The class *CustomUnit* can be used to define units that are necessary for a specific use case and that are currently not considered in the VEC.

The class *CompositeUnit* is used to define units that are created by the multiplication of other units (Association *CompositeUnit.factors*). For example, the Unit "Ohm per Metre" will be created with two instances of *SIUnit*. One *SIUnit Ohm* (without prefix and *exponent*) and one *SIUnit* Metre (without prefix and an *exponent* of "-1").

Mainly these *Units* are used by the class *ValueWithUnit* which can be either a *ValueRange* or a *NumericalValue*. A *ValueRange* defines a range of a value between *minimum* and *maximum.* A *NumericalValue* defines a single value with an optional *Tolerance.*

Note:

1. For the purpose of clarity, not all literals of all enumerations are displayed. If literals are omitted in the graphical representation a "..." is shown.
2. Currently the VEC does not support the definition of the same attribute value in different units at the same time (e.g. a single length value in m and ft at same time).

5.2.3 Extensibility with Custom Properties

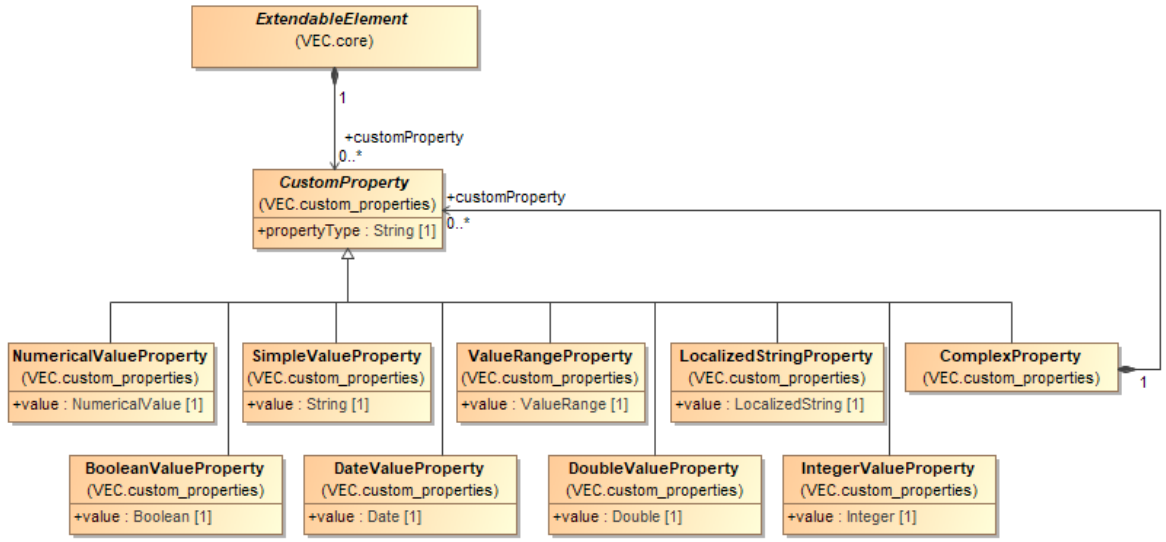


Figure 9: Extensibility with Custom Properties

CustomProperties have been introduced to the VEC as the dedicated extension mechanism. All subclasses of *ExtendableElement* class are extendable by providing the possibility to define *CustomProperties*. *CustomProperties* allow the definition and transport of almost any piece of data in an embedded way, for which the VEC does not define a different means of expression.

If a VEC entity shall be extended with multivalued property this is done by adding multiple *CustomProperties* with the same *propertyType.* In cases where a custom property value consists of a tuple of other values, a *ComplexProperty* can be used. This is especially useful, when the respective property is multivalued. An example for such a property / structure that is already represented in the VEC are *Colors* where each color consists of a referenceSystem and a key in that referenceSystem and each entity can have multiple colors (the same color in different reference systems).

Note: According to this data format specification it is strictly forbidden to store data within *CustomProperties* for which the VEC knows a special predefined way of expression. VEC-Files that do not obey to this rule are not compliant to this data format specification.

Note: Please refer to the detailed class description for information about which elements inherit from *ExtendableElement.*

5.2.4 Foundation Classes for Physical Properties

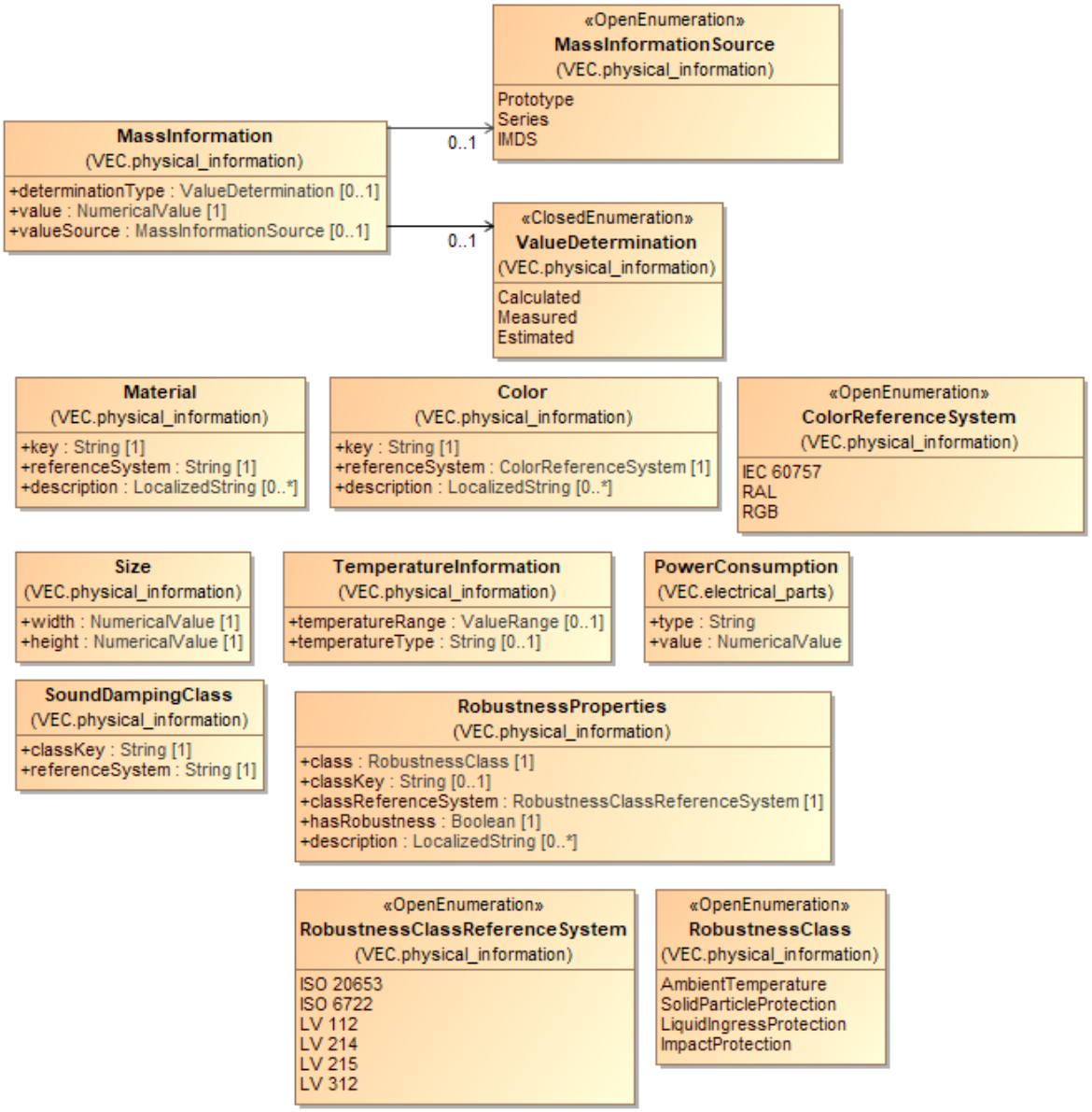


Figure 10: Foundation Classes for Physical Properties

The diagram displays the common foundation classes necessary for the description of physical properties of a part (e.g. material, mass and size). Most of these classes either define a type of their value (e.g. *TemperatureInformation.temperatureType*) or a reference system (e.g. *Color.referenceSystem*). In these cases attributes of these types normally have the multiplicity of [0..*] in order to define different values for different types (e.g. environment temperature, storage temperature, operating temperature) or to define the same value in different references systems (e.g. the same color in RAL, RGB or a company specific value).

5.2.5 Alias Identifications

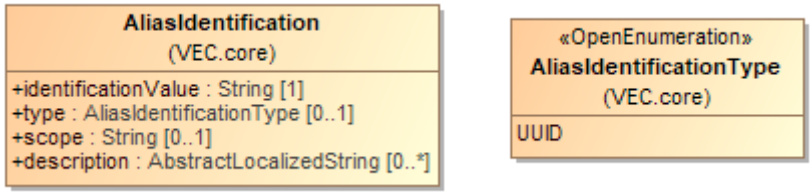


Figure 11: Alias Identifications

There are many applications where an element can have various technical and functional keys at the same time. The *AliasIdentification* provides a possibility to transfer these keys along with meta information about their type and scope.

5.2.6 Open and Closed Enumerations

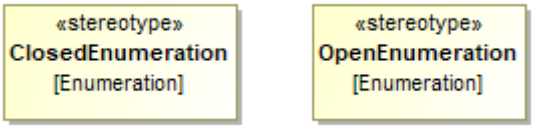


Figure 12: Open and Closed Enumerations

In the VEC model we distinguish between two kinds of enumerations:

1. Closed lists
2. Open lists

For a closed list the enumeration values are fixed. At the time of the UML model generation it is known, which values an attribute can take. There is no possibility to use other values later without changing the XML schema definition as the acceptable values are defined inherently by the model. An example would be the *PartVersion.primaryPartType* as it closely related to the existing *PartOrUsageRelatedSpecification-class*. Another example would be the *referenceNode* for *Locations* on *TopologySegments*. Since one *TopologySegment* only has one *StartNode* and one *EndNode* this is a typical *ClosedEnumeration*.

For an open list the allowed values for an attribute are not completely known at the point of the UML model generation or might change in future due to new technical developments. There may be a pre-defined set of values. But it must be possible to add further values in a later stage. An example might be the Gender of a Terminal which is currently restricted to *Male* and *Female*. In future new technologies might emerge, where the existing values are not enough. Motivation for this open list concept is, that for known values, standardized literals shall be used, but reading systems must be able to handle additional literals, since they can emerge quite quickly due to technology improvements.

The diagram shows the representation of open and closed enumeration lists as stereotypes.

During the UML to XML transformation, closed enumerations are transformed into XML "*xs:enumeration*". Open enumerations will be translated into XML „*xs:string*".

The VEC model defines for open enumerations a pre-defined list of commonly agreed values. The idea behind is, that these pre-defined values are used if applicable, and further values should be used only, if the pre-defined ones do not fit. However, this cannot be enforced by the XML schema. If an open enumeration is translated into "*xs:string*", it is not possible to check if the predefined values are used.

Therefore, in addition to the standard VEC XML schema, a further XML schema, called the "Strict VEC XML schema" is defined. In this schema, an open enumeration is transformed into XML "*xs:enumeration*", too. The enumeration values are restricted to the pre-defined ones. With this schema it is possible to check if a VEC used pre-defined values, only.

5.2.7 Open and Closed Pattern Restrictions

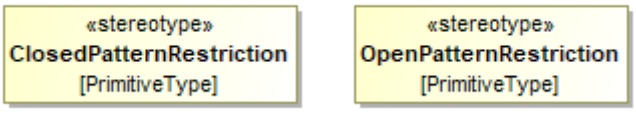


Figure 13: Open and Closed Pattern Restrictions

The concept of open enumerations mentioned before is useful in cases where a restriction to a limited amount of well-known values is necessary. However, the continuous evolution of the VEC has shown, that this approach is not feasible in all situations.

Therefore, the concept of pattern restrictions has been introduced. A pattern restriction is a regular expression that defines valid attribute values. Those restrictions can be used when a valid pattern is known, but the explicit enumeration of the values is not possible. For example, it is known that valid values for the coding of a connector are only strings with two digits of alphanumerical characters. To represent this as an enumeration would require 1296 enumeration literals.

In the UML model attributes that shall be restricted with a pattern are defined like regular class attributes with a custom *Primitive Type* as datatype (e.g. +coding: CodingName[1]). The *Primitive Type* is stereotyped either with *ClosedPatternRestriction* or *OpenPatternRestriction.* The semantic for open and closed is the same as for enumerations. *ClosedPatternRestrictions* are included in all schema versions, *OpenPatternRestrictions* only in the strict schema versions. The actual pattern of the restriction is defined as "class constraint" on the *Primitive Type.*

In the XML Schema the *Primitive Types* are translated as <*xs:simpleType*> based on *xs:string* with a *xs:pattern* restriction.

5.3 PDM Information

The content of this chapter describes the various aspects of PDM information contained in the VEC. PDM information is every information that is no actual content data, but all meta data about *ItemVersions* contained in the VEC.

Since most of the PDM information (product data management relevant information) is in principle equally relevant for both, parts and documents, the VEC defines the abstract superclass *ItemVersion* as anchor for the definition of PDM information.

5.3.1 Lifecycle Information

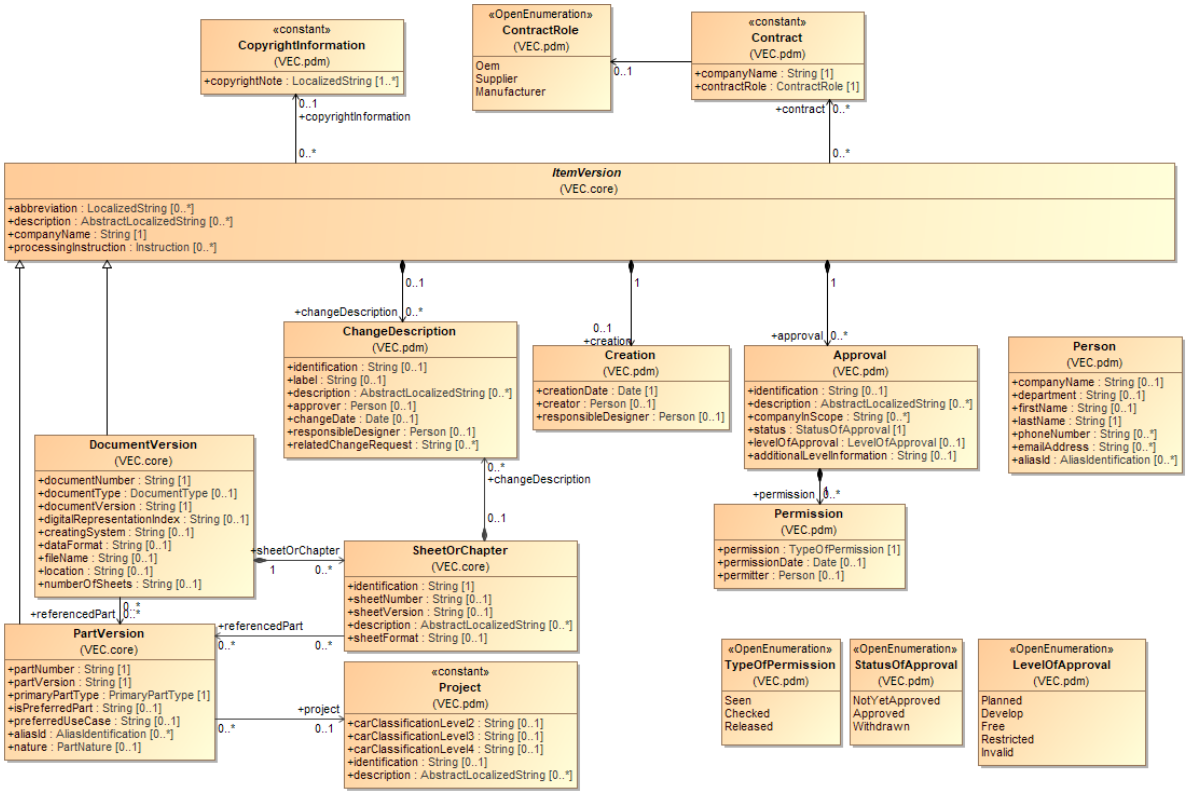


Figure 14: Lifecycle Information

An *ItemVersion* can carry information about the creation of the item. This means the creating timestamp and an optionally information regarding the creator or responsible person.

An *ItemVersion* normally has approval information and can be specified within the VEC by the relationship *ItemVersion.approval*. The class *Permission* allows the documentation of the persons involved in the approval-process.

The class *ChangeDescription* expresses the changes of an *ItemVersion* compared to its direct *predecessorVersion(s).*

The class *CopyrightInformation* enables the description of copyright information of *ItemVersions*. Instances of the class *CopyrightInformation* are assumed to be constant and so do not require a dedicated belonging to a versioned instance (*PartVersion* or *DocumentVersion*).

An *ItemVersion* can have contract information with company role information. Instances of the class *Contract* are assumed to be constant and so do not require a dedicated belonging to a versioned instance (*PartVersion* or *DocumentVersion*).

A *PartVersion* can reference a *Project* in order to describe for which *Project* it is relevant. Instances of the class *Project* are assumed to be constant and so do not need to have a dedicated belonging to a versioned instance (*PartVersion* or *DocumentVersion*).

5.3.2 Baselines

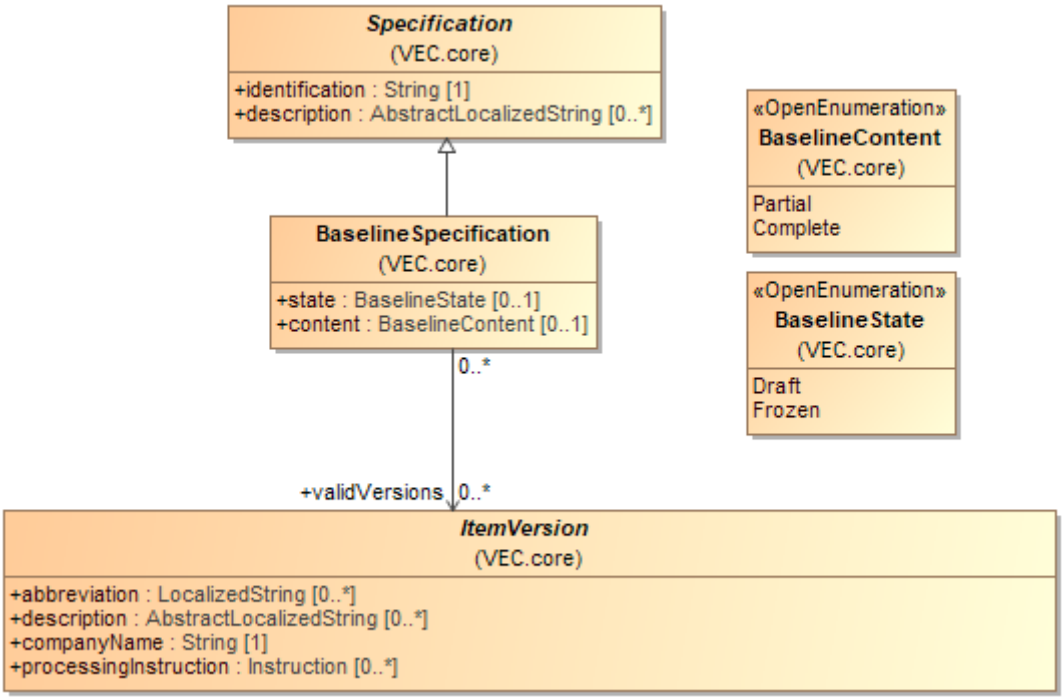


Figure 15: Baselines

A *BaselineSpecification* defines a set of *ItemVersion*s (*Document-* and *PartVersions*) that relate to each other in a certain way e.g. all parts and documents in their specific versions that contributed to a specific manufactured result.

Baselines are a standard mechanism to support change, release and configuration management and are often used throughout the whole product life cycle. In the VEC, baselines themselves are *Specifications* and thus content of a *DocumentVersion*, which is equivalent to the fact that they are versioned information themselves.

In a product development for wiring harnesses, the content of a baseline is growing over a period (e.g. system schematics are created before the definition of the physical wiring harness). In this process, items are created and added to the baseline (based on other items from the baseline). Additionally, items in the baseline can be changed, following the rules of the respective development process. Such a baseline has the *BaselineState.Draft.* At a well-defined point in time, the content of baseline is checked

and released. After this, the baseline cannot be changed without creating a new version of the baseline itself. Such a baseline has the *BaselineState.Frozen.*

Baselines are normally defined for a certain scope (e.g. a specific vehicle, a specific product). During the product development it is possible, that a baseline is incomplete. A reason for this might be, that not all development artefacts (e.g. wiring harnesses) are defined in this phase of the development process even though all artefacts are required for the final product. Such *BaselineSpecifications* have a *Partial content.*

5.3.3 Item History

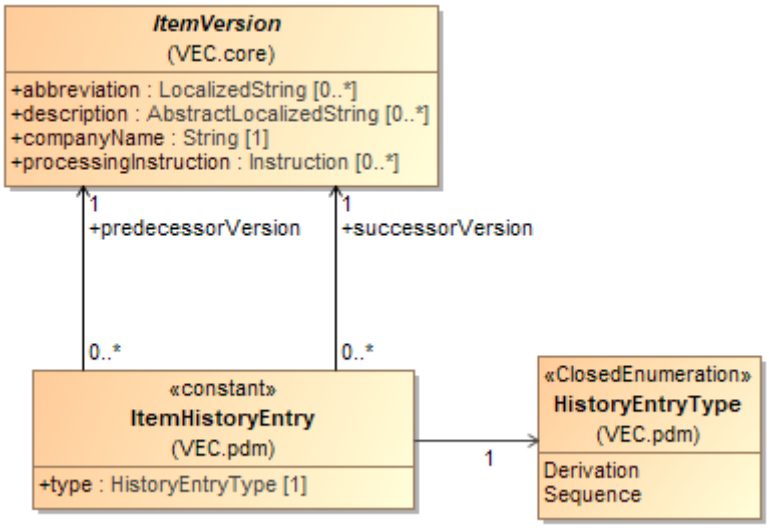


Figure 16: Item History

In a development process items (parts or documents) are often refined incrementally, so one *ItemVersion* originates from another. The class *ItemHistoryEntry* enables the description of predecessor / successor relationships between *ItemVersions*. The description of both linear and branching relationships is possible. The relationship can be classified in order to distinguish between two cases:

- The successor *ItemVersion* is an instance of the same item (*type = "Sequence"*)
- The successor *ItemVersion* is an instance of a new but somehow similar item (*type = "derivation"*).

5.3.4 Item Equivalence

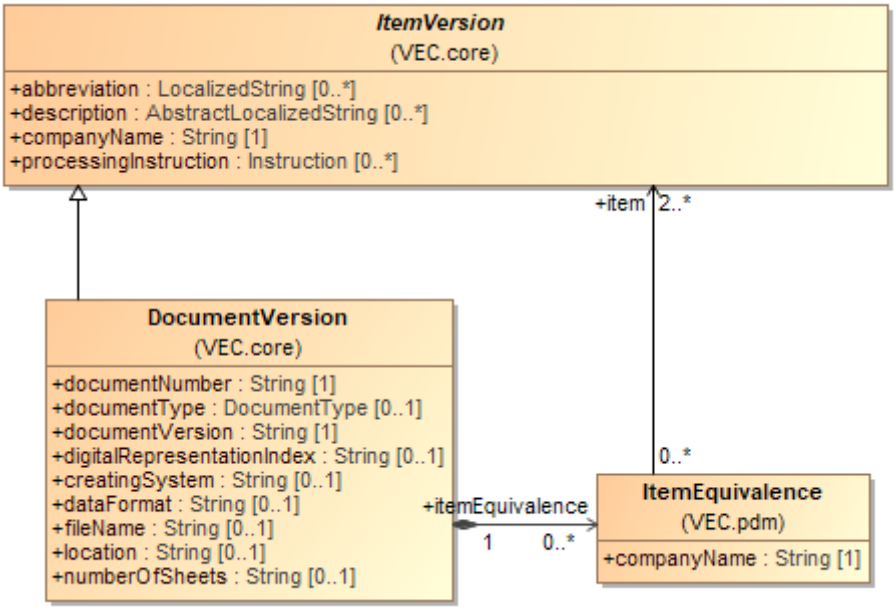


Figure 17: Item Equivalence

The class *ItemEquivalence* enables the description of equivalence relationships between two or more *ItemVersions*. The standard use case is to relate *ItemVersions* defined in the context of one company (numbering system) to *ItemVersions* defined in the context of another company.

An extended use case might be that the differences are not only limited to different numbering systems, but even to technical attributes. As the equivalence statement might be very subjective the corresponding *companyName* must be specified.

5.4 General Component Data

The description of components and their properties is quantitatively the largest part of the VEC model. The following sections explain the basic methods for describing components and defining the information that applies to all types of components.

The properties of certain component types are defined in the next chapter.

5.4.1 Description of Parts

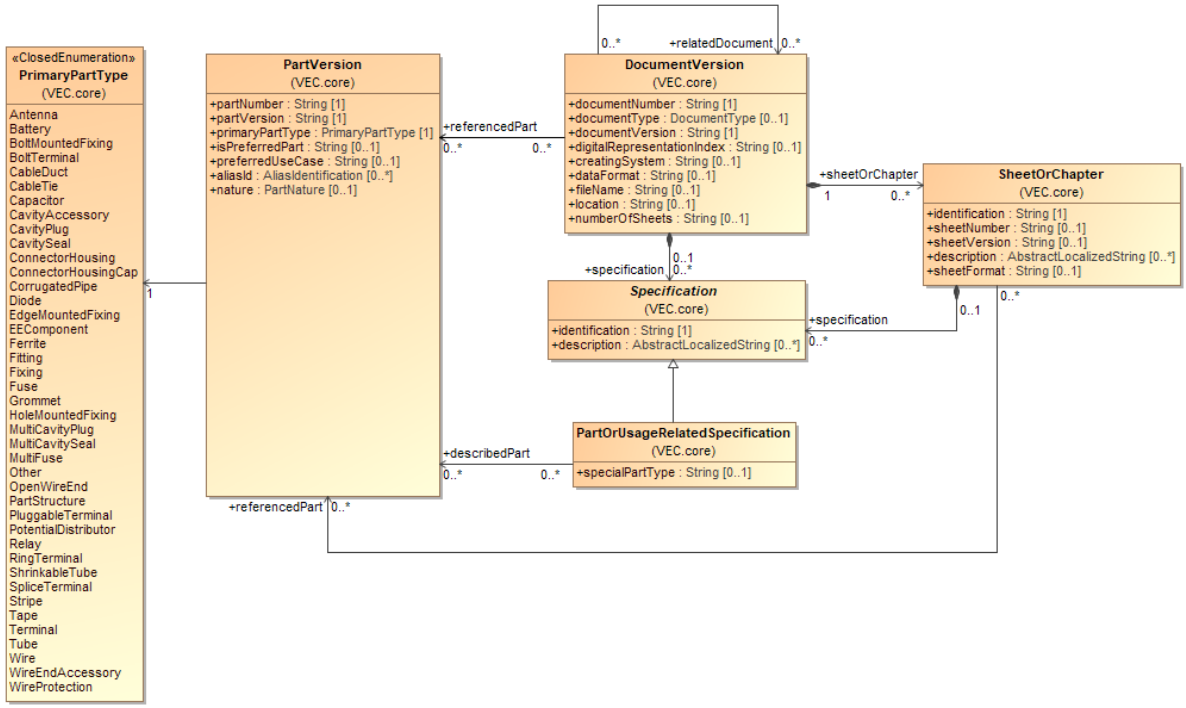


Figure 18: Description of Parts

In traditional data formats the diverse types of parts (e.g. connector housing, terminal, wire) are differentiated with individual subclasses for each type inheriting from the type corresponding to *PartVersion*. These subclasses define all specific information of the component type (e.g. the cavities of a connector). Due to the great diversity of components used in wiring harnesses a different approach is implemented in the VEC.

The general technical / physical properties of a component are described by *PartOrUsageRelatedSpecifications* (and its subclasses). For each aspect of a component an independent subclass of *PartOrUsageRelatedSpecification* is defined (e.g. a *ConnectorHousingSpecification*, a *TerminalSpecification*). This approach allows to describe a *PartVersion* by more than one *PartOrUsageRelatedSpecifications* (Association *PartOrUsageRelatedSpecification.describedPart*). It offers the possibility to extract information, which is shared between component types, into separate specifications so that the structural definition can be reused without having to modify the inheritance tree (e.g. *GeneralTechnicalPartSpecification*). Furthermore, some components cannot be mapped onto such a strict type hierarchy (e.g. a connector housing used as an inliner might have a fixing bolt for vehicle body in order to keep it in place). Representing this by an inheritance tree would result in a huge variety of different classes. With the concept of *PartOrUsageRelatedSpecifications* a component like the mentioned connector housing can be easily described by using a *ConnectorHousingSpecification* for the connector housing aspects and a *FixingSpecification* for the fixing aspects. However, most parts have a primary characteristic. For example, the mentioned connector housing will most probably only be used if a connector is necessary. This primary characteristic is defined by the

PartVersion.primaryPartType. The *PrimaryPartType* always refers by a naming convention to a corresponding *PartOrUsageRelatedSpecification*. If the *PrimaryPartType* is ABC, the corresponding *Specification* is named *ABCSpecification* (e.g. *ConnectorHousing* and *ConnectorHousingSpecification*). The *PrimaryPartType* serves the purpose of defining the primary characteristic of a part. It is not required to describe every *PartVersion* in a specific VEC file with a corresponding *PartOrUsageRelatedSpecification,* if the information is not required in the context of the VEC file. For example, in the context of geometry data exchange it might be perfectly valid to have *PartVersions* with *PrimaryPartType=ConnectorHousing* that are only associated with a *PlaceableElementSpecification* (and/or a *GeneralTechnicalPartSpecification*) while the *ConnectorHousingSpecification* is not provided.

In some cases, there might be a need to specify attributes of a part that is not related to any of the existing subtypes of *PartOrUsageRelatedSpecification.* (e.g. it might be relevant to describe a screw nut used as an accessory for a connector housing). In this case the class *PartOrUsageRelatedSpecification* is instanced and the relevant attributes are added as *CustomProperties*.

Since the *PartOrUsageRelatedSpecification* is a subclass of *Specification,* which is a sub element of a dedicated *DocumentVersion,* all information defined by it can be related to a precise development state.

The relationship *PartOrUsageRelatedSpecification.describedPart* might be seen as redundant to the relationship *DocumentVersion.referencedPart* but considers the fact that there are use cases, where only information on the level of *PartVersion*s and *DocumentVersions* is exchanged, but not on the very detailed level of *PartOrUsageRelatedSpecification*s. In addition, it is possible, that several *PartVersions* are described by a single *DocumentVersion* containing a couple of specifications, each of them describing a single *PartVersion* (e.g. a drawing with a major part and its accessory parts).

5.4.2 General Technical Part

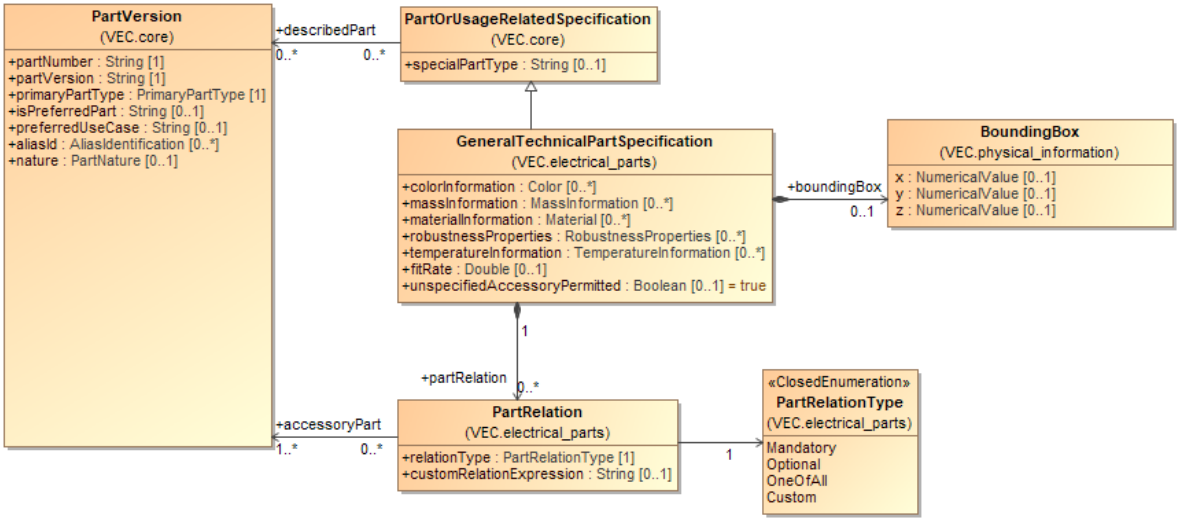


Figure 19: General Technical Part

The class *GeneralTechnicalPartSpecification* is intended to aggregate common technical attributes that are relevant for most kinds of parts (e.g. *massInformation* or information about valid temperature environments).

5.4.3 Supplementary Parts

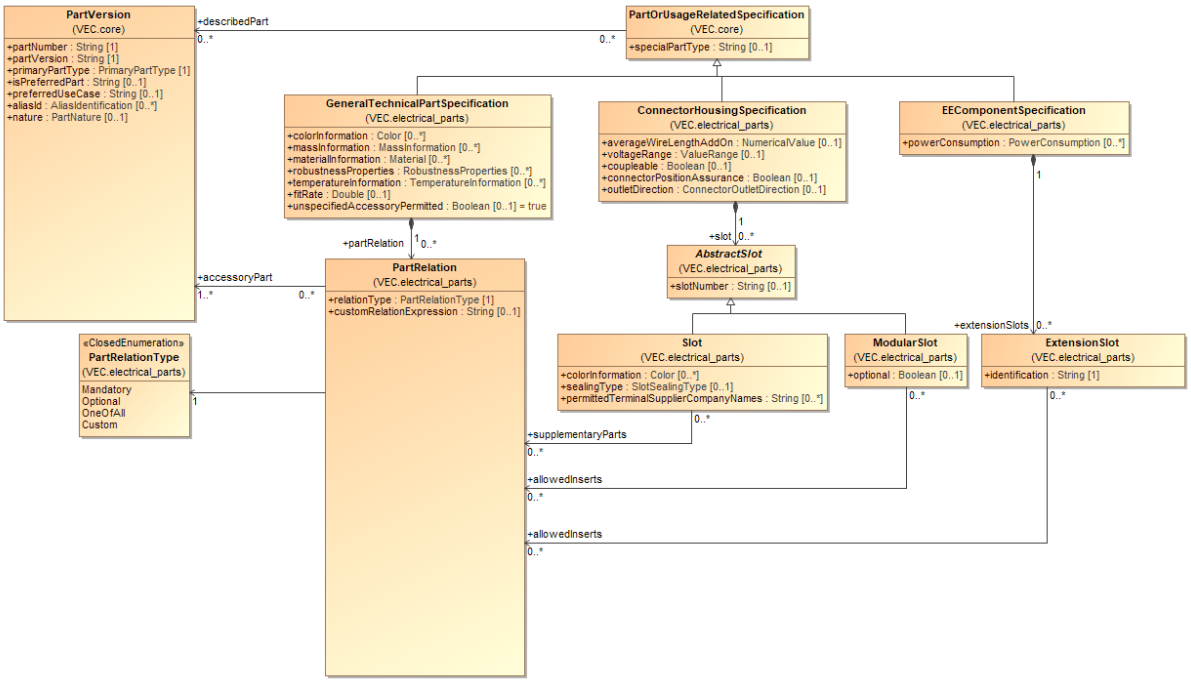


Figure 20: Supplementary Parts

With the *PartRelations* other *PartVersions (those)* can be defined as supplementary parts for the *PartVersions* that are described by the containing *GeneralTechnicalPartSpecification (this)*.

The semantic is, that if *this PartVersion* shall be used in a correct way in a product, those referenced *PartVersions* can or shall be added to the product as well (depending

on the used *PartRelationType*). That can be for example caps, bars or levers in case of a connector.

The *PartRelation.relationType* defines how the set of referenced *PartVersions* of one *PartRelation* must be interpreted. For details see the description of class *PartRelation.*

The association via the *PartRelation* just defines what other components may be used together with this component. It does not further specify the location of the usage or its function. However, in some cases a more detailed specification of the location is needed, e.g. if the supplementary part is required for a specific slot in a connector housing. In such a case, the *PartRelation* is specified by the *GeneralTechnicalPartSpecification* and may be referenced by its specific usage location (currently from a *Slot, ModularSlot* or *ExtensionSlot*).

5.4.4 Placeable Elements

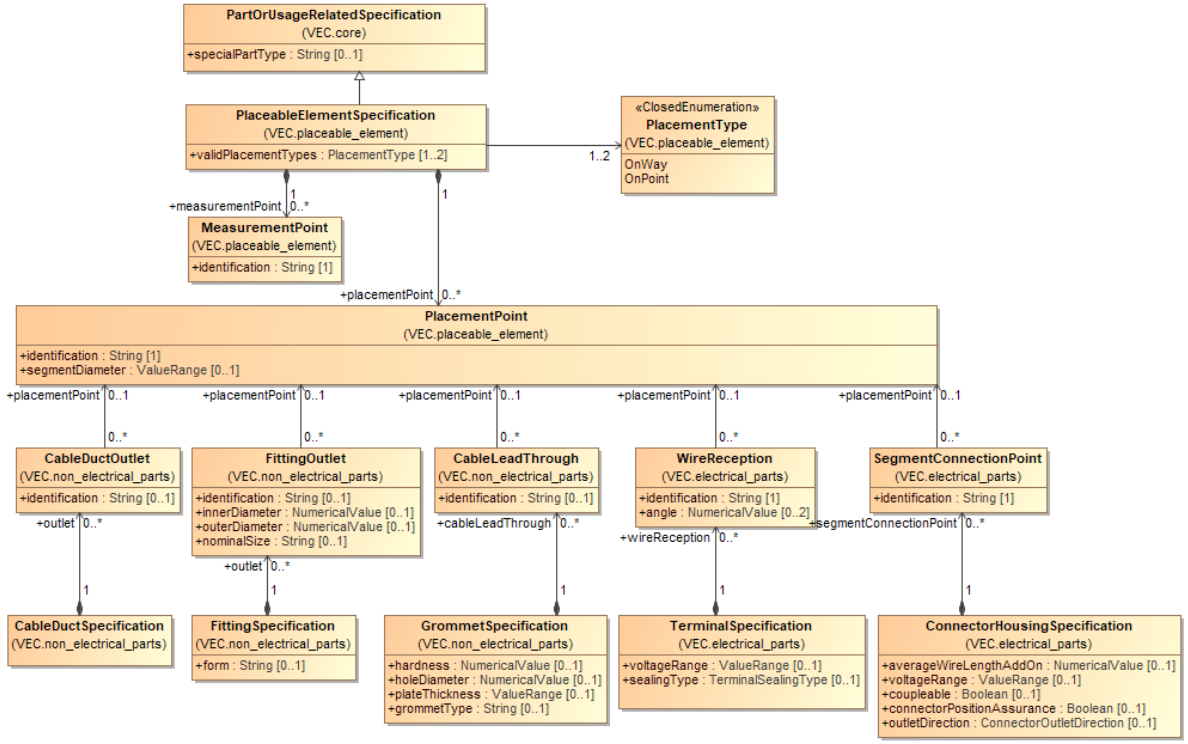


Figure 21: Placeable Elements

In a wiring harness many components can be placed directly onto the topology (e.g. connectors, fixings). To specify the aspect of placeability the VEC defines a *PlaceableElementSpecification*. Every component to be placed directly onto the topology must have a *PlaceableElementSpecification* in addition to its regular type specific *PartOrUsageRelatedSpecification* (e.g. *ConnectorHousingSpecification*).

A *PlaceableElementSpecification* defines at minimum the *validPlacementTypes* for the component (e.g. a connector can be placed on a point, a tape can be placed on a path in the topology).

In addition, the *PlaceableElementSpecification* can define important points of the component, *MeasurementPoints* and *PlacementPoints.*

MeasurementPoints are locations on the component that can be used to define a dimensioning. To make a *MeasurementPoint* locatable by a person more information is needed. This information is normally provided with a drawing of the component. The *identification* can be used to find the *MeasurementPoint* on the drawing (if it is labelled). To make a *MeasurementPoint* locatable by a system, the external mapping mechanism can be used.

PlacementPoints are locations on the component, where it can be attached to the wiring harness. Therefore, it can define a *segmentDiameter* for which it is usable. To locate a *PlacementPoint* on the component, the same approach as for *MeasurementPoints* can be used.

Often these *PlacementPoints* have additional properties depending on the component type. In these cases, the corresponding *PartOrUsageRelatedSpecification* redefines these elements, with a reference to the *PlacementPoint* which represents them (e.g. the *SegmentConnectionPoint*). However, a *PlacementPoint* can be defined without such a corresponding element, if no additional information is necessary.

5.4.5 Coordinate Systems of Components

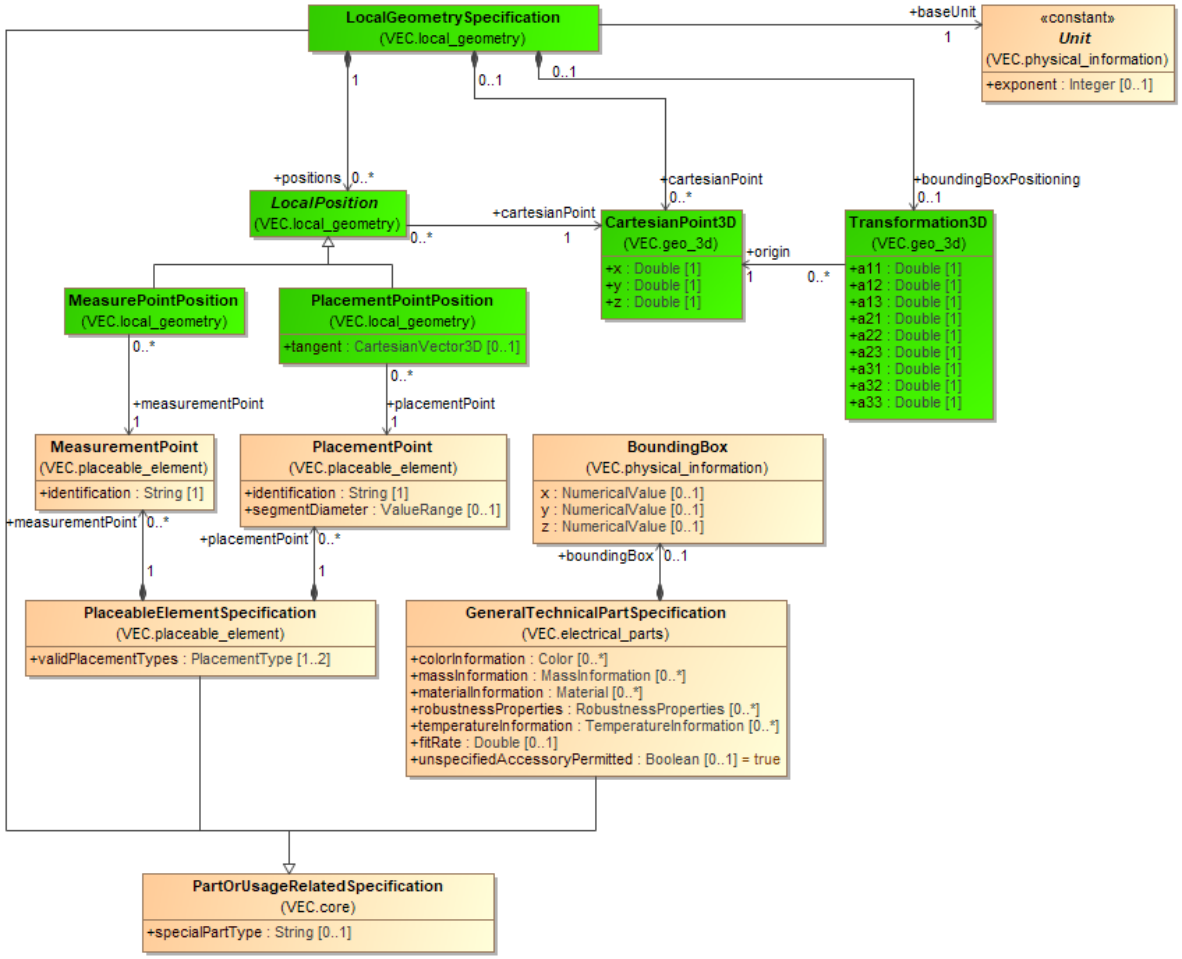


Figure 22: Coordinate Systems of Components

A *LocalGeometrySpecification* defines a mapping between the entities of the VEC and the 3D model of a component. The 'Local' in the name refers to the fact that all definitions within this specification are 'local' to the 3D model of a specific component (a component in a library, not in a specific usage).

The coordinate system of a component is defined by its 3D model, which often follows certain guidelines or conventions (e.g. the coordinate systems of connectors have their origin in the centre of cavity 1). The *Transformation3D* of an *OccurrenceOrUsageViewItem3D* (see section 3D-Geometry) defines how the 3D model of the component shall be transformed in order to represent the correct positioning of its usage. This transformation is defined in regard to the origin and the axis of the component's coordinate system.

The *BoundingBox* specifies a box that has the extent to completely enclose the volume of the component. It is defined to have one corner in the origin of its coordinate system and to grow along the respective axes in positive direction. This means, that the origin of bounding box is on a corner of the represented volume, the origin of the 3D model of the component is at some arbitrary point within the represented volume. As a result, the bounding box will most likely not enclose the component's 3D model in "usage space" when placed with the same transformation as the original 3D model. To overcome this problem, the *LocalGeometrySpecification* defines a *Transformation3D* as *boundingBoxPositioning,* which puts the generally defined bounding box into the coordinate system of the component. The transformed bounding box is then placed into the "usage space" with the same transformation as the 3D model of the component.

In addition, the *LocalGeometrySpecification* also defines *LocalPositions* of important points of the component. The position of these points is defined within the coordinate system of the component. Positions can be defined for:

- *PlacementPoints:* Points where a TopologySegment is attached to the component (e.g. a Bundle / Segment Connection Point of a Connector).
- *MeasurementPoints:* Significant points on the component on to which dimension can be defined (e.g. for QA purposes).

5.4.6 Part Substitutions

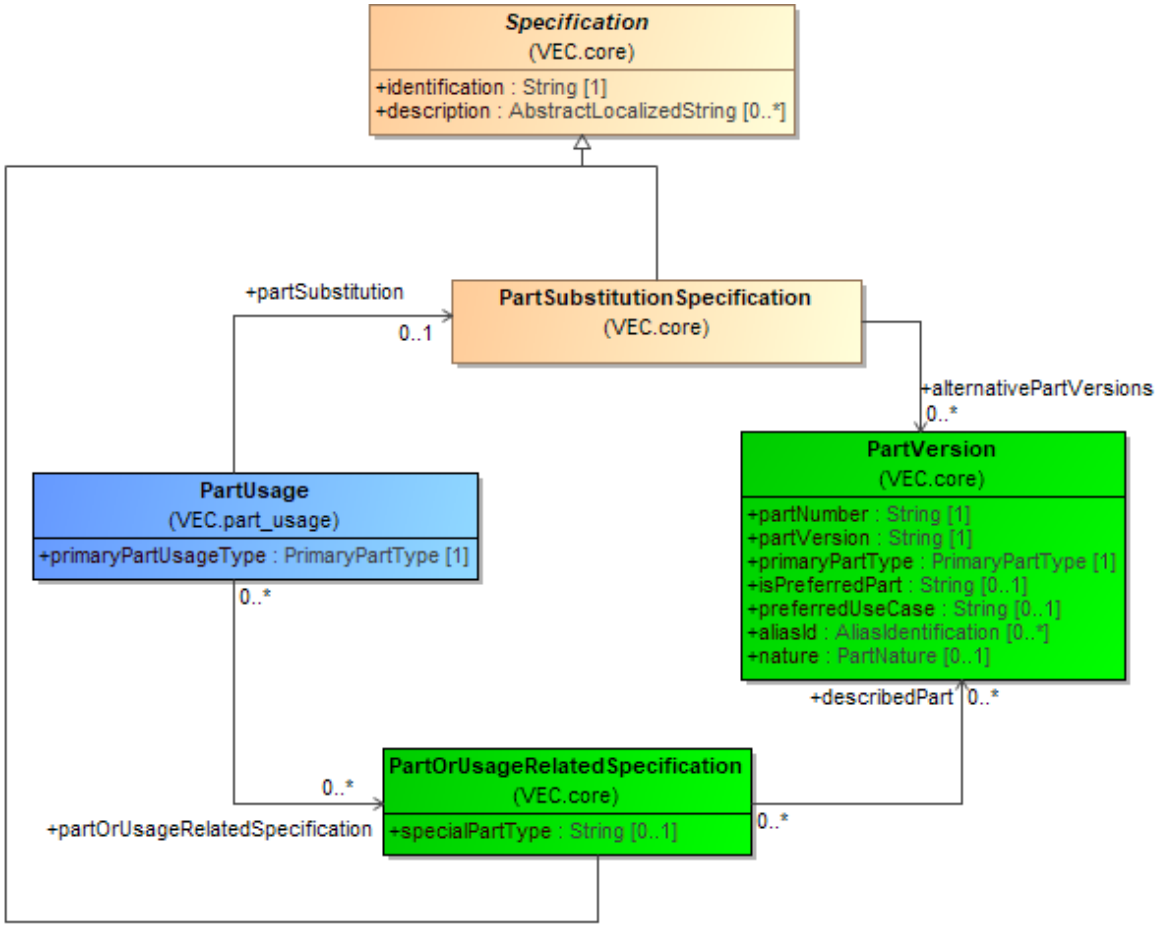


Figure 23: Part Substitutions

A *PartSubstitutionSpecification* defines a set of *PartVersions* that can be used alternatively, due to an incomplete specification for the 150% product description. For a concrete wiring harness only one valid *PartVersion* remains. The selection logic for valid *PartVersions* is not included in the VEC. It is NOT valid to use a *PartSubstitutionSpecification* for *PartVersions* with identical properties that just have different *PartNumbers* in different contexts (e.g. multi supplier topics). For these cases an *ItemEquivalence* shall be used.

A *PartSubstitutionSpecification* can be used e.g. for tubes or ring terminals, where a part of the specification is known at design time, but not yet the complete specification. E.g. for tubes the required inside diameter is not known at design time, since it depends on the bundle diameter of a specific configuration.

In order to represent a component instance that utilizes such a *PartSubstitutionSpecification* a *PartUsage* is necessary. The *PartUsage* is the element in the VEC which defines instances of components, where a specific *PartVersion* is not yet known. It references one or more *PartOrUsageRelatedSpecifications* to describe the known properties of the component. If the *PartUsage* references an additional *PartSubstitutionSpecification* the set of valid *PartVersions* can be further constrained.

5.4.7 Conformance to Requirements

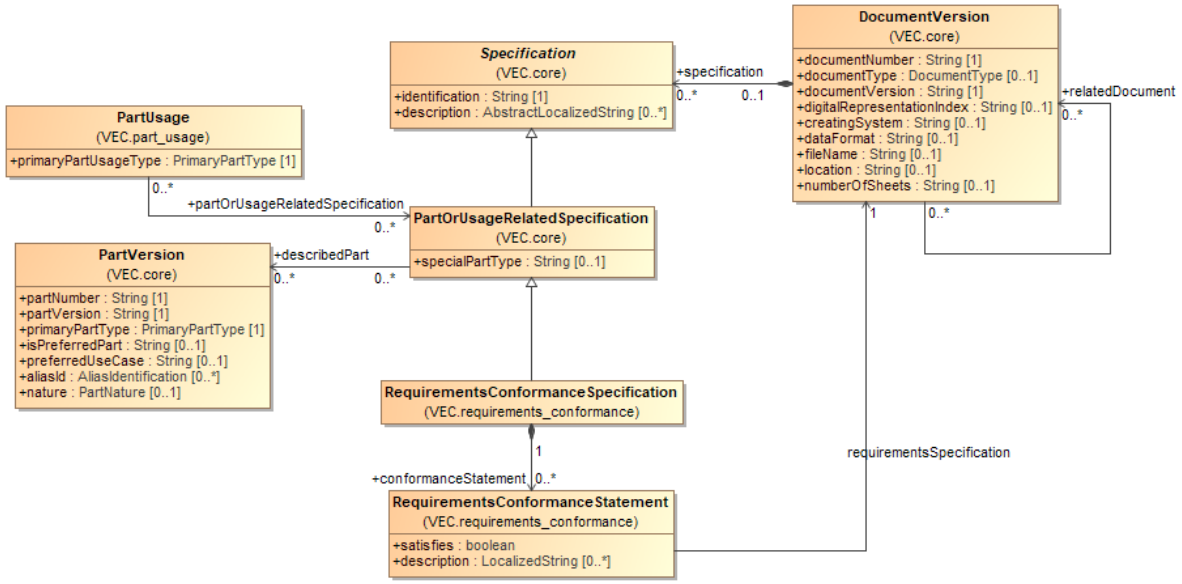


Figure 24: Conformance to Requirements

During the life cycle of a given PartVersion manifold situations are existing, where the conformance of that PartVersion to a set of requirements must be demonstrated or is required to be known. To represent the result of such a conformance test, the VEC offers the *RequirementsConformanceSpecification* and the *RequirementsConformanceStatement.*

As the *RequirementsConformanceSpecification* is a *PartOrUsageRelated-Specification,* it can be used to express conformance for a set of *PartVersions* or *PartUsages*. The *DocumentVersion* in which the *RequirementsConformance-Specification* is contained, normally represents the original document by which the conformance was approved (e.g. a type rating).

The actual conformance is expressed with *RequirementsConformanceStatements.* It can be used to document a successful or explicitly failed conformance test. The requirements for which the state is valid, are referenced as a *DocumentVersion* via the *requirementsSpecification* association.

If a *RequirementsConformanceStatement* is omitted for a *PartVersion*, this does not imply any conformance information at all. The *RequirementsConformanceStatement* can be missing because the component has never been tested according to the specification, or it has been tested, but the information was not transferred within this specific VEC instance, or the information is just not available in a digitally analysable form.

The VEC does not impose any restrictions on the kind of requirements specifications for which conformance can be expressed. This can be for example:

- A standard (company or public) like an ISO or a DIN.
- A definition of requirements in the sense of a specification sheet.

- Requirements that allow certain handling procedures (e.g. suitability for production automation)

The VEC does not impose any restrictions on how the conformance must be demonstrated. This can be done with well-defined testing procedure, a manual assessment, an audit, a type examination or anything else. Normally the requirements specification itself defines how conformance to it must be demonstrated.

5.5 Component Characteristics

5.5.1 Wire

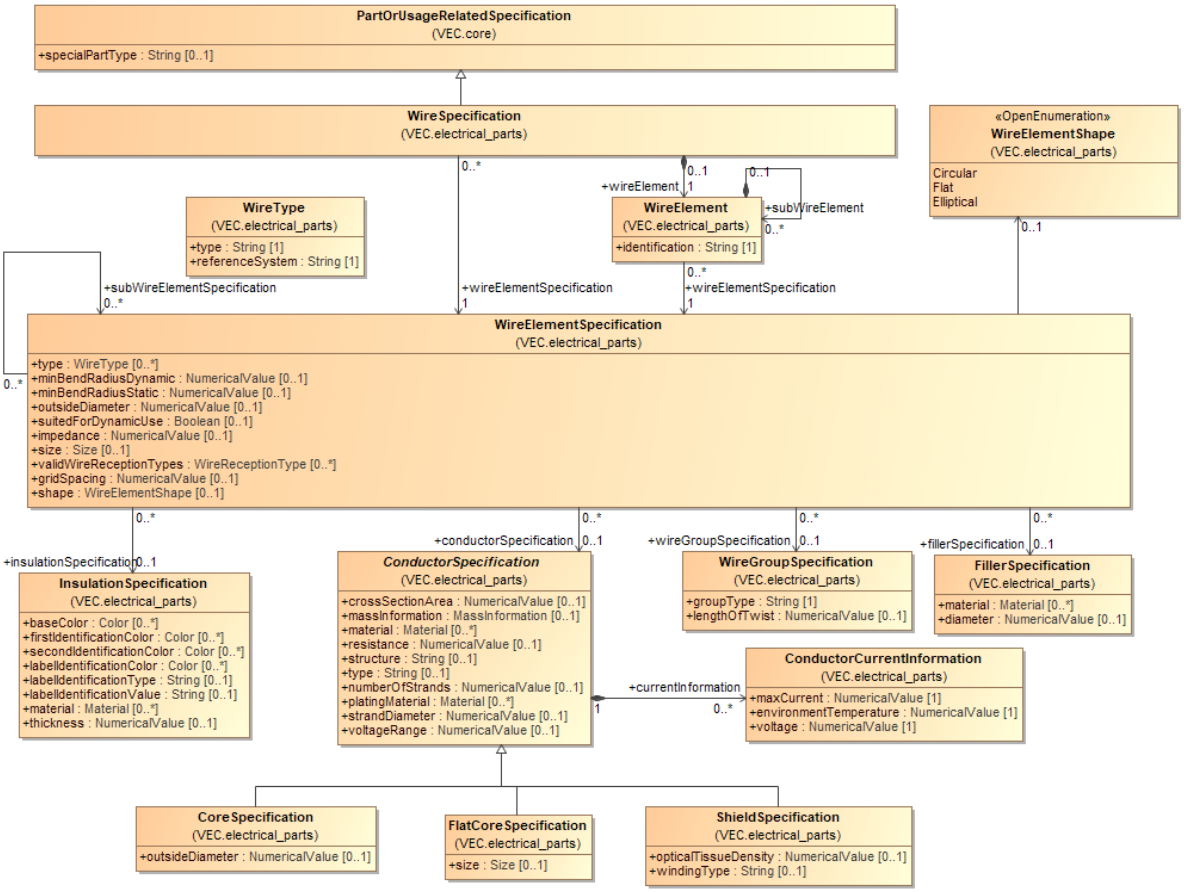


Figure 25: Wire

In the VEC wires are defined through a *WireSpecification,* regardless of their type. This means, for all types of wires (e.g. normal single core, multi core or coax wires) the same structure is used to describe them. Since a wire can be a hierarchical structure the actual definition of the structure is delegated to a *WireElementSpecification*. A *WireElementSpecification* can define a certain wire element and more complex structures by referencing the appropriate *subWireElementSpecifications.*

This model structure is required, because some *WireElements* can exist as individual parts and as an element of one or more complex wires with the same technical properties (e.g. a FLRY-0.75 wire can be used standalone or as part of a multi core or twisted wire). In order to allow the reuse of such elements, the structure of a wire element is defined with *WireElementSpecification*s which can be shared and reused

between different other *WireElementSpecifications* and *WireSpecifications*. This means a *WireSpecification* references the root *WireElementSpecification* that describes its structure, while the *subWireElementSpecifications* can be used by different *WireSpecifications* at the same time.

In order to allow an unambiguous identification of a particular *WireElementSpecification* within the context of a *WireSpecification*, the *WireSpecification* defines a list of *WireElements* for each *WireElementSpecification* that belongs to the hierarchy of the wire. The *WireElement* defines the *identification* of a *WireElementSpecification* within the context of a wire.

A *WireElementSpecification* can reference an *InsulationSpecification*, a *CoreSpecification,* a *ShieldSpecification,* a *FillerSpecification* and/or a *WireGroupSpecification* in order to describe its technical details. These aspects are separated into individual *Specifications* in order to allow the reuse of them. For example, the *CoreSpecification* of a FLRY-0.75 is the same for a group of wires, regardless of their insulation color. In turn the *InsulationSpecification* of a blue & green FLRY wire might be the same for a group of wires, regardless of their cross-section area.

When creating the hierarchy of *WireElementSpecifications* for a wire the representation with the minimal amount of *WireElementSpecifications* shall be used. This means that a single core shall be represented by a single *WireElementSpecification* with an *InsulationSpecification* and a *ConductorSpecification* and not with individual hierarchical *WireElementSpecifications* for the insulation and the conductor.

In most cases, a *WireElement* has a physical representation within the wire. However, there are cases were the *WireElementSpecification* is just a group of *WireElementSpecifications* with no real physical manifestation. For example, a twisted pair wire, consists of two single core *WireElementSpecifications* and a parent *WireElementSpecification* that just defines the type of twist. Note: All *Specification* in this diagram, where no superclass is displayed inherit directly from *Specification.*

5.5.2 Terminals

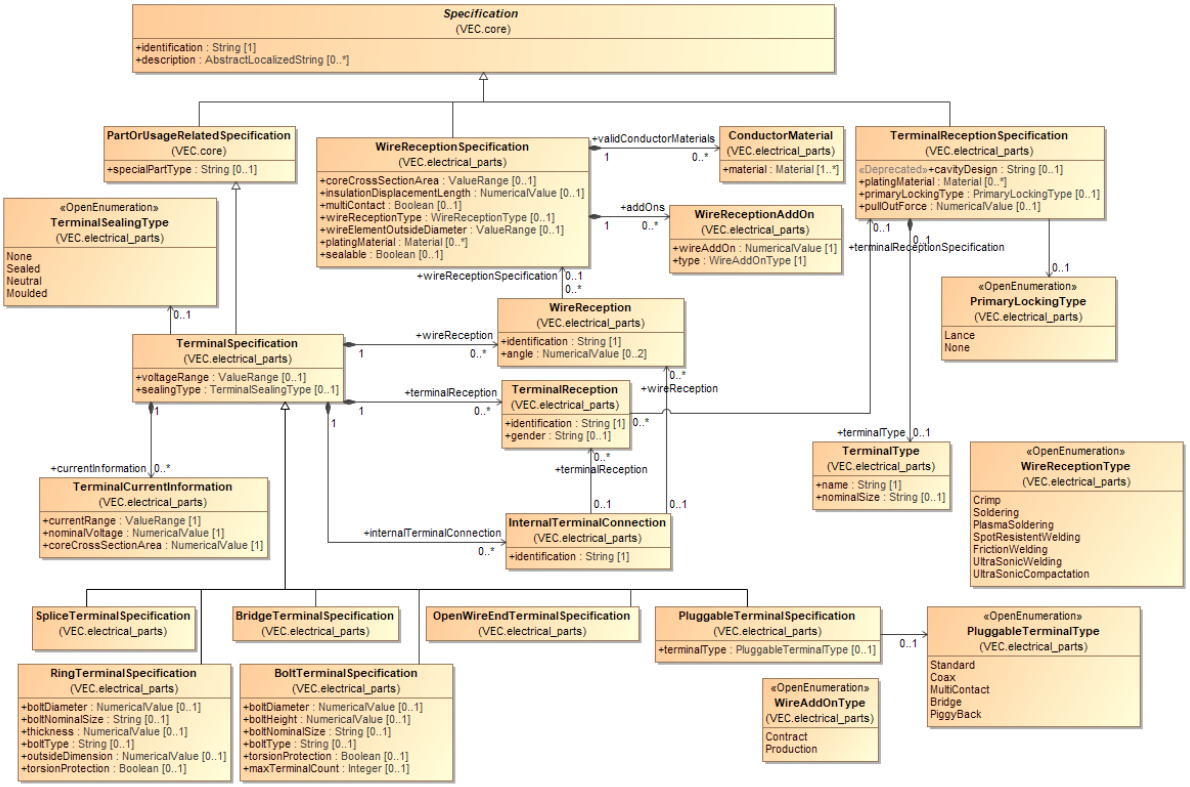


Figure 26: Terminals

A *TerminalSpecification* can define *WireReceptions* and *TerminalReceptions* and on that basis *InternalTerminalConnections*. A *WireReception* can reference a *WireReceptionSpecification* in order to provide technical details about the wires which can be accommodated. A *TerminalReception* can reference a *TerminalReceptionSpecification* which provides technical details about compatible cavities and mating-terminals. An *InternalTerminalConnection* states which *WireReceptions* and *Terminal-Receptions* are electrically connected. This is especially important in case of a coax terminal.

5.5.3 Wire End Accessory

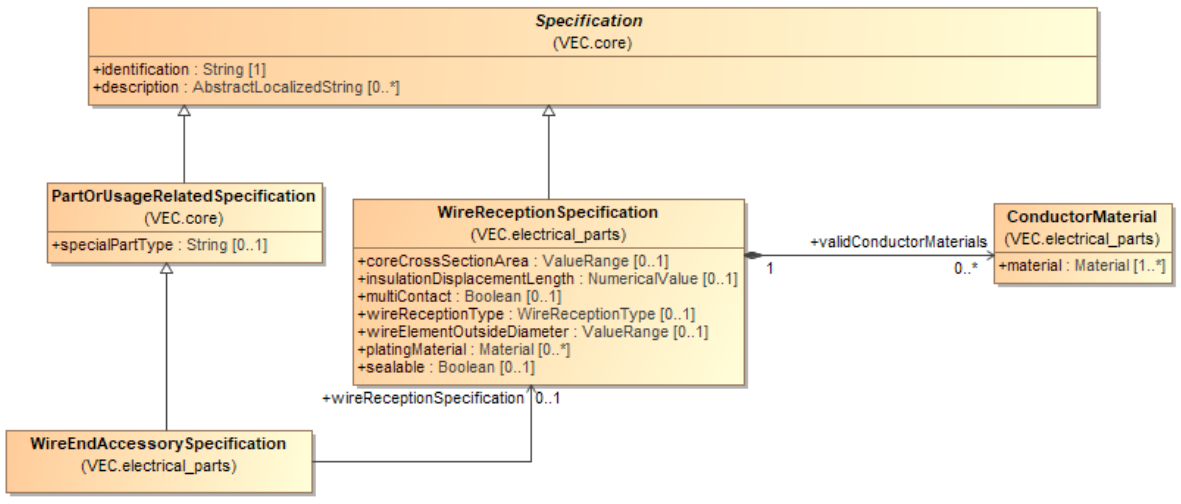


Figure 27: Wire End Accessory

WireEndAccessorySpecifications are describing parts that are used for a treatment of a wire end before the actual terminal is attached to the *WireEnd* (e.g. a ferrule).

5.5.4 Terminal Pairing

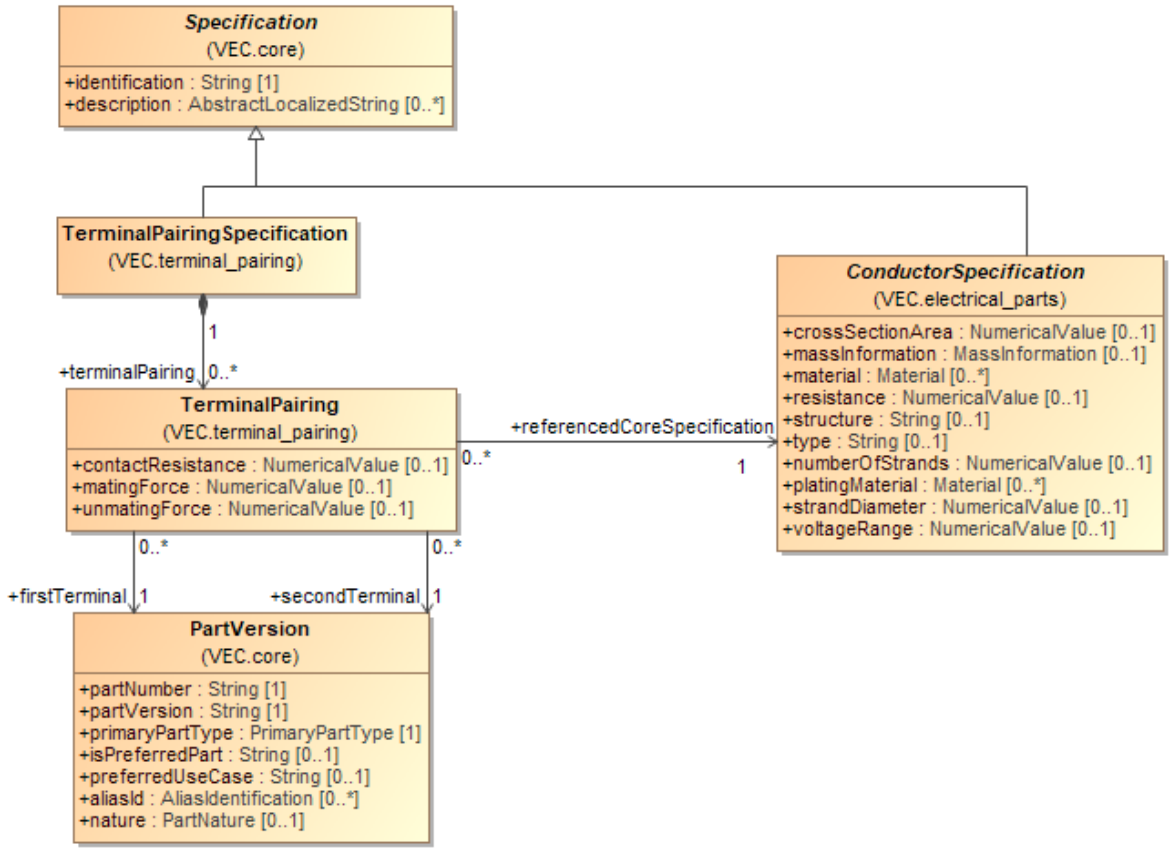


Figure 28: Terminal Pairing

A *TerminalPairingSpecification* is a container for various *TerminalPairing*. A *TerminalPairing* is an aspect of part master definition of terminals and represents a specific pair of terminals pluggable to each other. It specifies properties of that terminal pair in combination with a specific *ConductorSpecification.*

5.5.5 Connector Housings

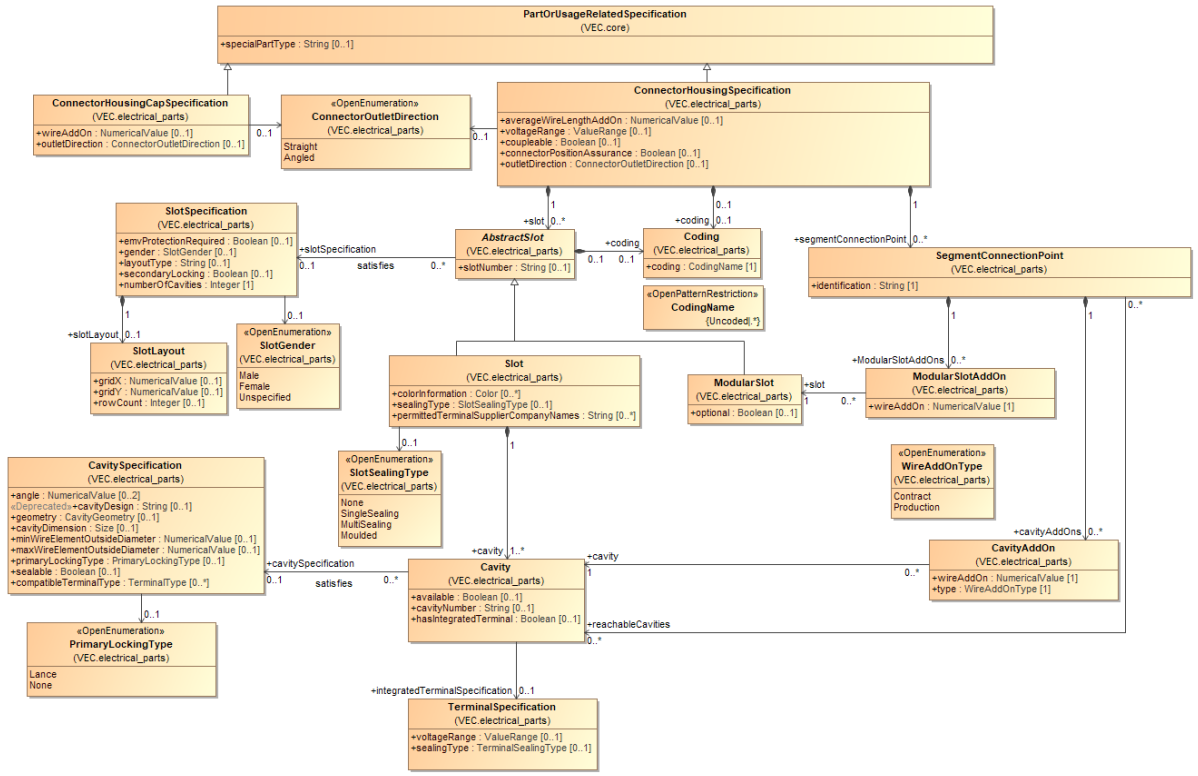


Figure 29: Connector Housings

A *ConnectorHousingSpecification* defines the technical properties of a connector housing or similar items (e.g. modular connectors, contact modules, connector shells). A connector is normally plugged into a mating connector in order to create connectivity (this is in most cases electrical connectivity, but not limited to it). The actual electrical connection is normally established by terminals, mated to each other, which are placed in cavities. The VEC supports two primary types of connectors: normal (conventional) connectors and modular connectors.

A conventional connector defines a fixed number of *Cavities*, which are grouped into a fixed number of *Slots.* A modular connector defines a geometrical shape into which another component (a contact module) can be placed. The cavities are then defined by the used contact module. The combination of both is possible as well.

Therefore, the C*onnectorHousingSpecification* consists of *AbstractSlots*, which either can be *Slots* or *ModularSlots* in an arbitrary combination.

A *Slot* is used if the cavities are part of the connector housing. Therefore, a *Slot* consists of *Cavities.* A *Cavity* is defined in detail with a *CavitySpecification*.

A *ModularSlot* is used if the slot is just a geometrical shape into which different inserts can be placed. The insert itself is again defined by a *ConnectorHousingSpecification.* So, it is possible to create recursive structures. A *ModularSlot* can reference more than one insert, because different variants of e.g. contact modules that fit into this particular *ModularSlot* might exist.

A *SegmentConnectionPoint* can be used if a connector housing has more than one entry point for wires (segments). The *SegmentConnectionPoint* is used to define which *Cavity* is reachable through which entry point. In the latter this is a necessary information to support auto routing for connectors with more than one *SegmentConnectionPoint*.

5.5.6 Cavity Mapping

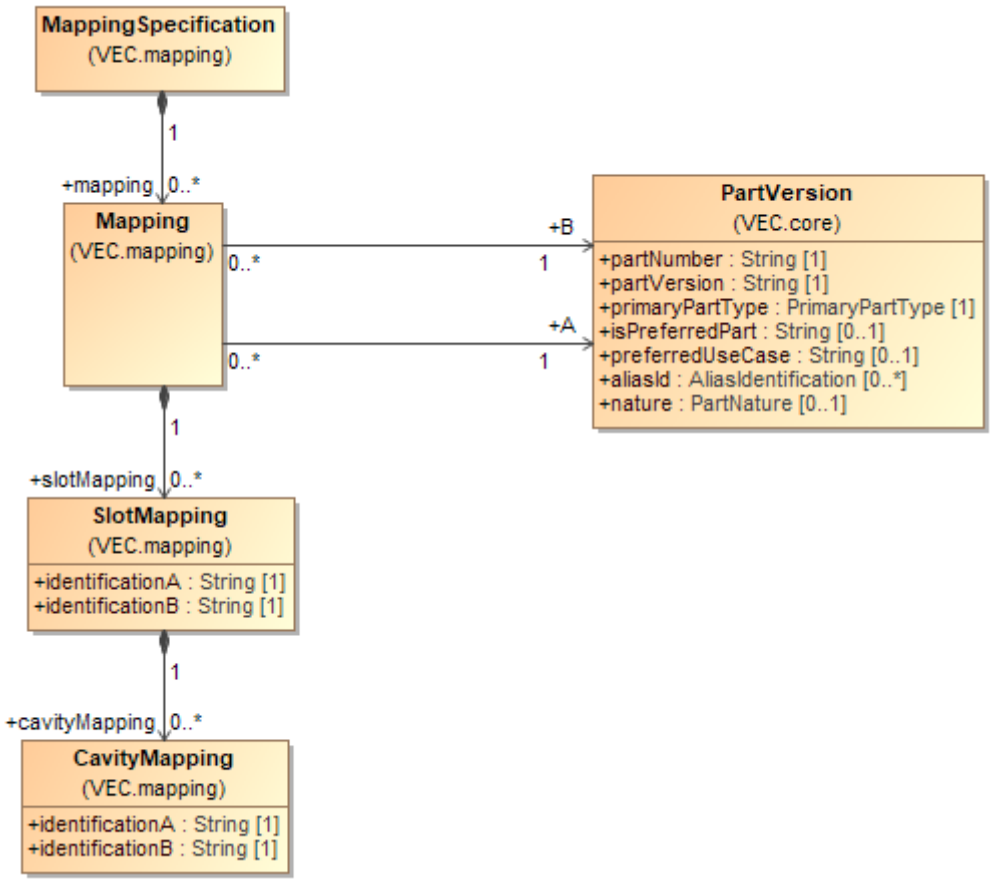


Figure 30: Cavity Mapping

A *MappingSpecification* defines the mapping of cavities in one connector housing onto the cavities in another connector housing. In order to create a compact data structure, the mapping is not done with direct references to *Cavities,* but with *Mapping* elements referencing two *PartVersions* for which the mapping is valid.

The *Mapping* element contains *SlotMappings* and *CavityMappings* to define which *Cavity* of one side gets plugged into which *Cavity* of the other side. The mappings are based on functional keys (e.g. cavity number), not technical keys like *ID/IDREF*, to ensure flexibility when partitioning the data into different VEC files.

For a valid *Mapping,* the referenced *PartVersions* shall represent *ConnectorHousings*. The identificationA & B of the SlotMapping / CavityMapping shall have correspondent *Slot.identifcation & Cavity.identification* in the respective part master definition of the referenced *PartVersions.*

5.5.7 Cavity Seals and Cavity Plugs

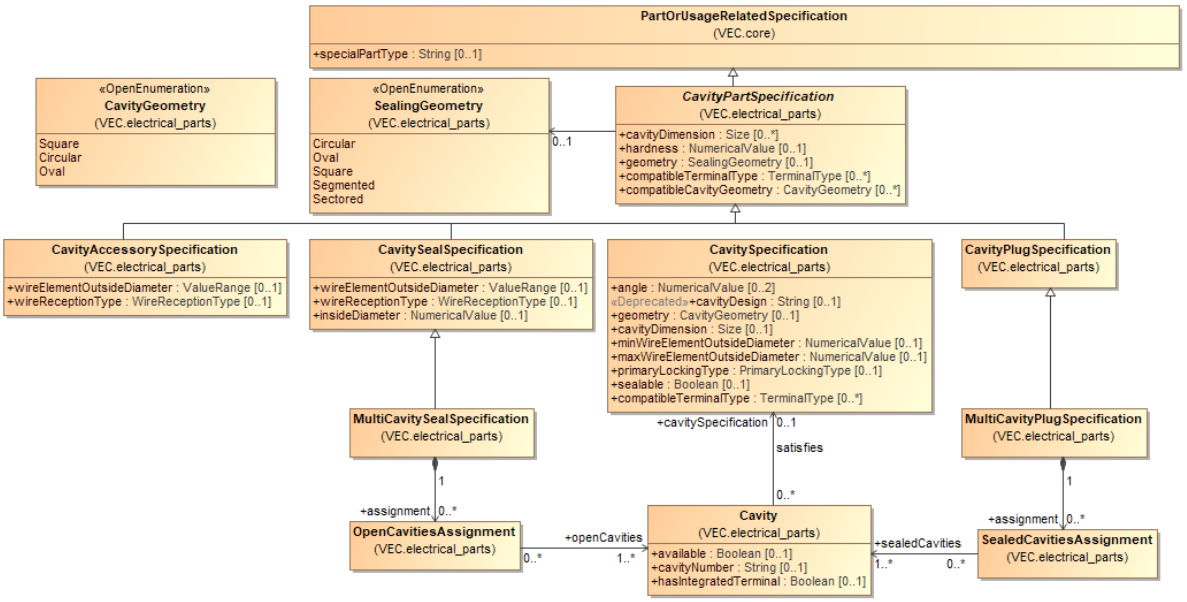


Figure 31: Cavity Seals and Cavity Plugs

The diagram displays the *PartOrUsageRelatedSpecifications* for Cavity seals and plugs. Both are used to seal *Cavities*. A cavity plug is used for a single cavity that has no wire in it, a cavity seal is used to seal a single cavity together with a terminal and a wire (single wire seal).

The *MultiCavitySeal* and the *MultiCavityPlug represent parts* which are sealing more than one cavity at the same time (multi wire seal). For more details see the description of the corresponding classes.

5.5.8 Wire Protections

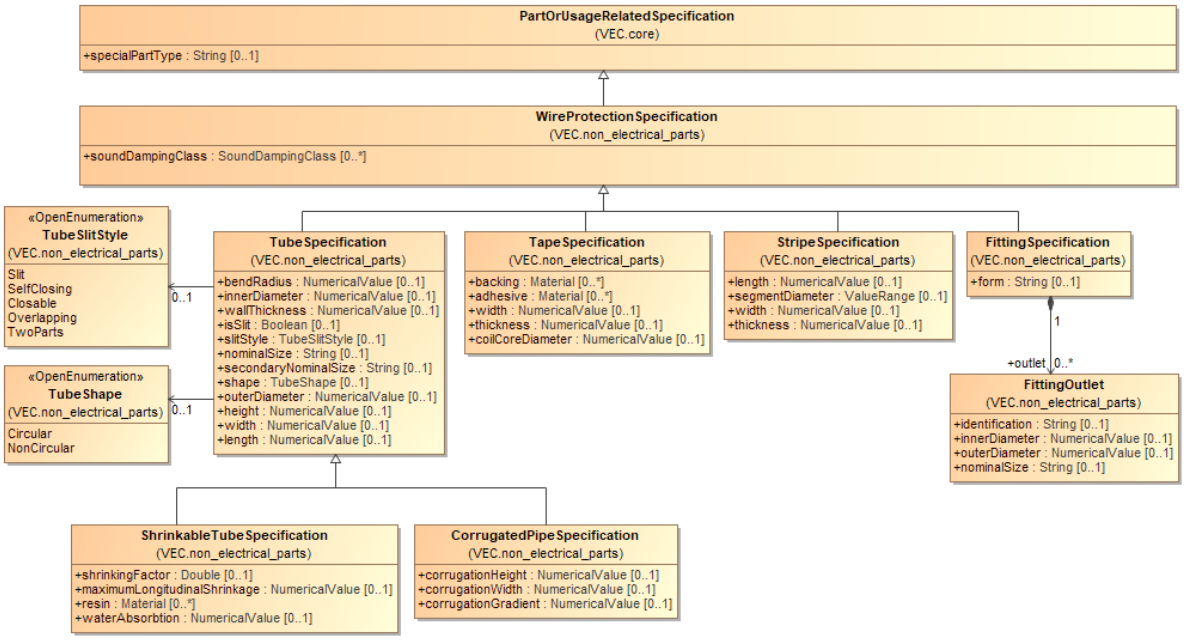


Figure 32: Wire Protections

The diagram displays the type hierarchy for *WireProtectionSpecifications.* These *PartOrUsageRelatedSpecifications* are used to describe the technical aspects of wire protections like tapes, tubes, fittings and stripes.

5.5.9 Fixings, Cable Ducts, Cable Ties and Similar

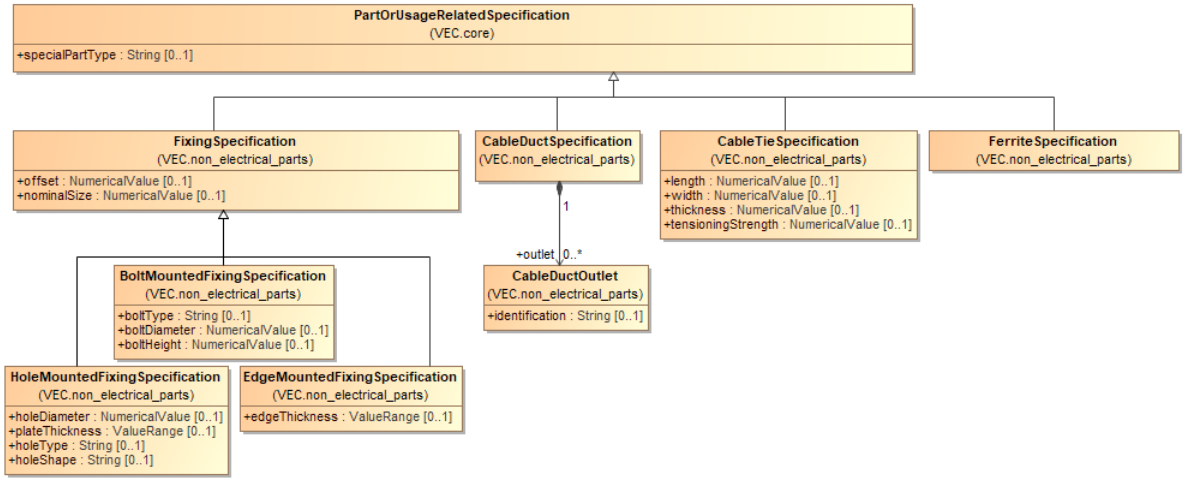


Figure 33: Fixings, Cable Ducts, Cable Ties and Similar

The diagram displays the type hierarchy for parts that are usually mounted onto the topology of a wiring harness. These *PartOrUsageRelatedSpecifications* are used to describe the technical aspects of components like fixings and cable ducts.

5.5.10 Grommets

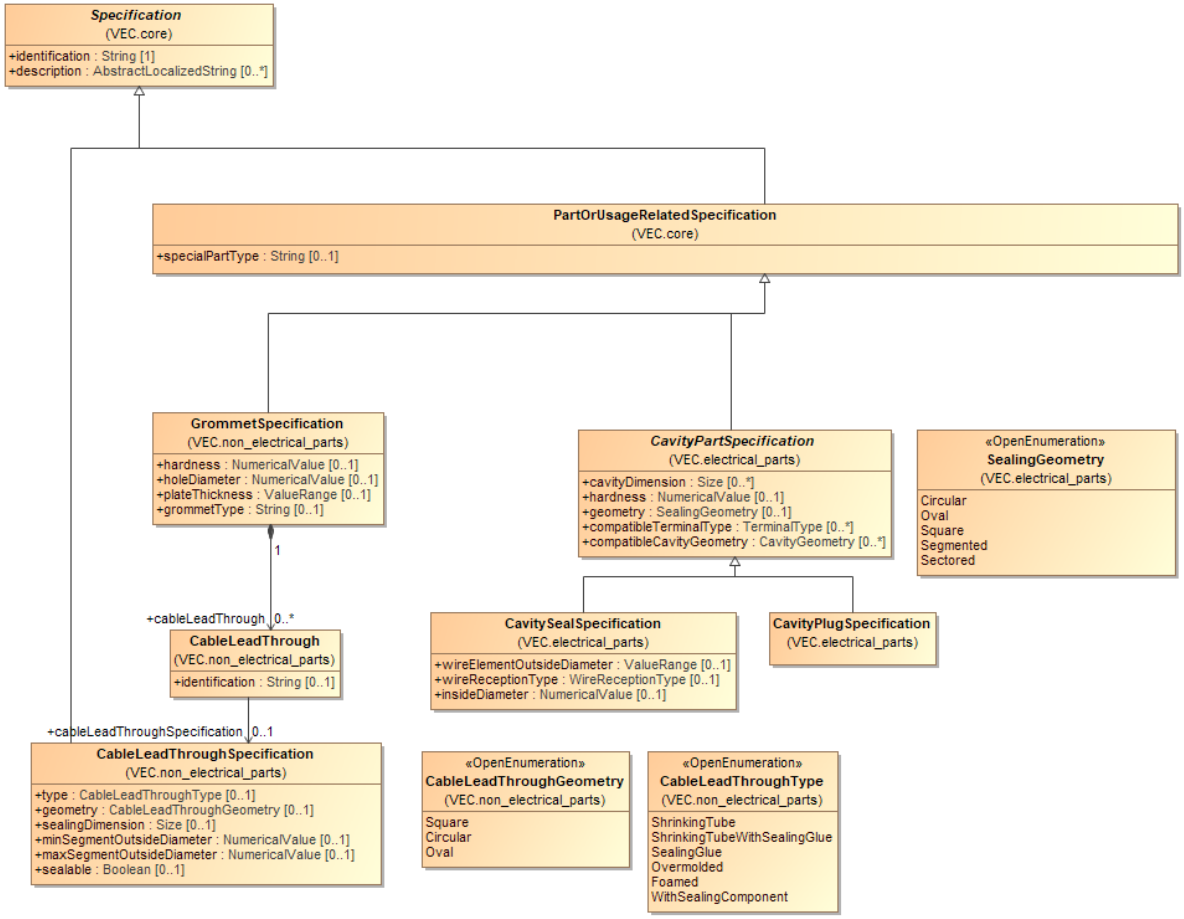


Figure 34: Grommets

This diagram displays the relevant classes for grommets. Basically, grommets belong to the same category of parts like fixings and cable ducts. Grommets are mounted onto a harness topology and are used to guide cables safely, and in some cases sealed, through openings, for example in the car body.

There are cases, where additional sealing of a cable lead through is not done with some kind of manufacturing steps (e.g. Foamed), but with additional sealing components. These are common case e.g. in high voltage applications where one lead through per wire is used. Those are often sealed with an additional single wire seal. In the VEC those seals are represented by a *CavitySeal*- or *CavityPlugSpecification*. Even if the name "Cavity..." seems awkward in the combination with grommets & cable lead throughs, the properties remain the same.

The *CableLeadThroughSpecification* includes all attributes that are necessary to calculate appropriate seals.

5.6 EE-Components

5.6.1 EE-Components

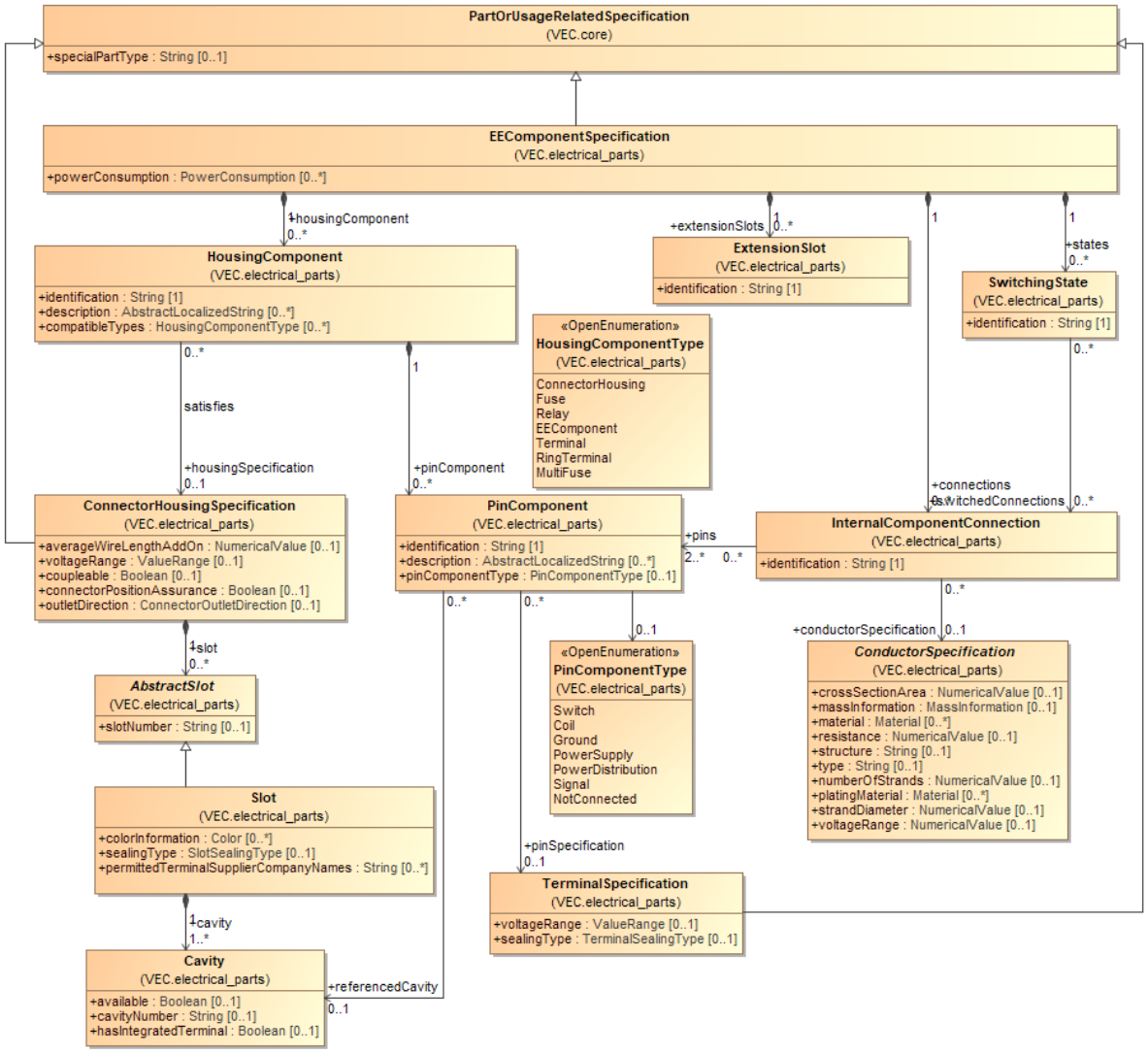


Figure 35: EE-Components

An *EEComponentSpecification* is used to describe the technical aspects of an EE-component. An EE-component is an electrical component to which the wiring harness is attached to (e.g. an ECU, a sensor, a relay, an antenna, a battery, a fuse). All EE-components have in common that they have a certain electrical interface (e.g. to which the wiring harness is attached). To describe this interface, the *EEComponentSpecification* is used.

An *EEComponentSpecification* can define a couple of *HousingComponents.* A *HousingComponent* is one interface to which other components can be attached (e.g. the wiring harness). To describe a *HousingComponent* satisfyingly two aspects are necessary: its geometrical shape and its electrical behavior. In order to describe the geometrical shape of a *HousingComponent* it can reference a *ConnectorHousingSpecification,* so the same concept is used to describe a connector in the wiring harness and in an EE-component. The electrical behavior of a

HousingComponent is defined by several *PinComponents.* A *PinComponent* references a *Cavity* to define its location in the geometrical shape of the *HousingComponent.* To define its physical properties a *PinComponent* references a *TerminalSpecification.* These properties are defined inherently by the material and the type of the pin. It is a common case that some of the *PinComponents* (sometimes all) of a single *HousingComponent* are referencing the same *TerminalSpecification*. The definition of properties regarding the behavior of a *PinComponent* (e.g. defined by the software deployed on an ECU) is explained in the diagram "Pinning Information & Pinning Variance".

5.6.2 EE-Component subclasses

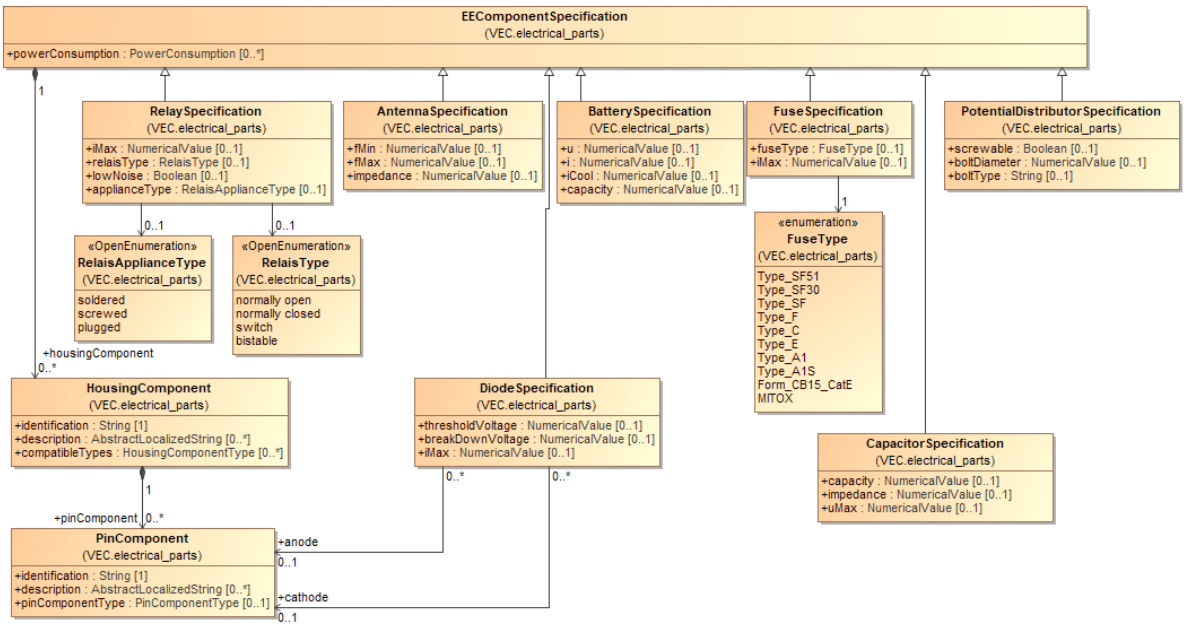


Figure 36: EE-Component subclasses

In order to define additional aspects of an EE-component there are some subclasses of the *EEComponentSpecification*. This diagram displays these subclasses.

5.6.3 Multi Fuse

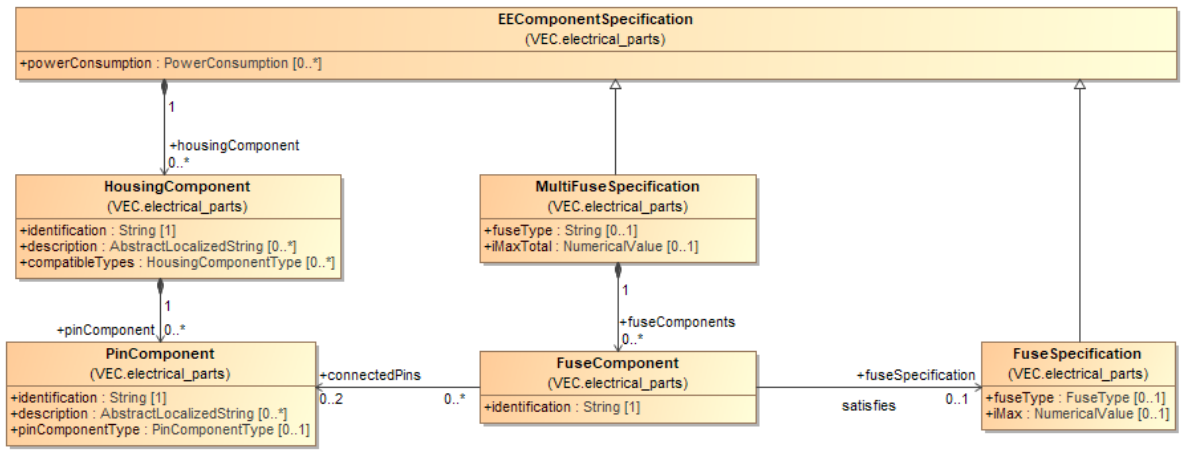


Figure 37: Multi Fuse

A multi-fuse is a metal structure that combines several fuses with a common power feed in a fixed layout. It is used to distribute a power supply to different fuse paths.

5.6.4 Pinning Information & Pinning Variance

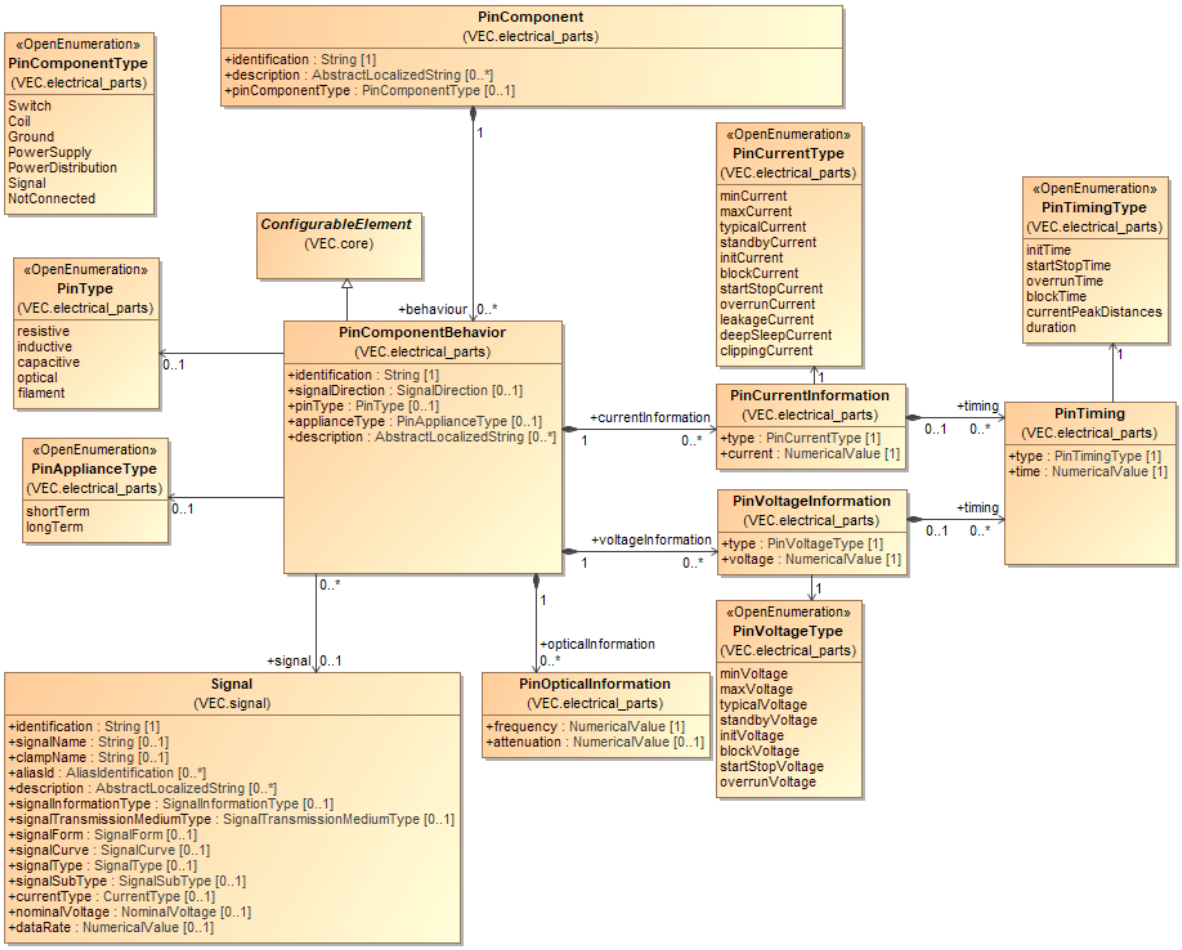


Figure 38: Pinning Information & Pinning Variance

The electrical behavior of a *PinComponent* can be fixed for a certain EE-component but does not have to be fixed. For example, in the case of an ECU, the behavior depends on the deployed software.

Therefore, the electrical behavior of a *PinComponent* is not defined directly and fixed in the *PinComponent*. In fact, a *PinComponent* can define *PinComponentBehaviors*, which are *ConfigurableElements.* This means the actual valid *PinComponentBehavior* for a *PinComponent* can be determined when the necessary variant information is available. When no variant information is necessary (the *PinComponent* has a fixed behavior) then the *PinComponent* just defines a single *PinComponentBehavior.*

The *PinCurrentInformation* and *PinVoltageInformation* can be used to describe the characteristic curves for different types and times of currents and voltages.

5.7 Instances of Components

5.7.1 Instantiation of Components

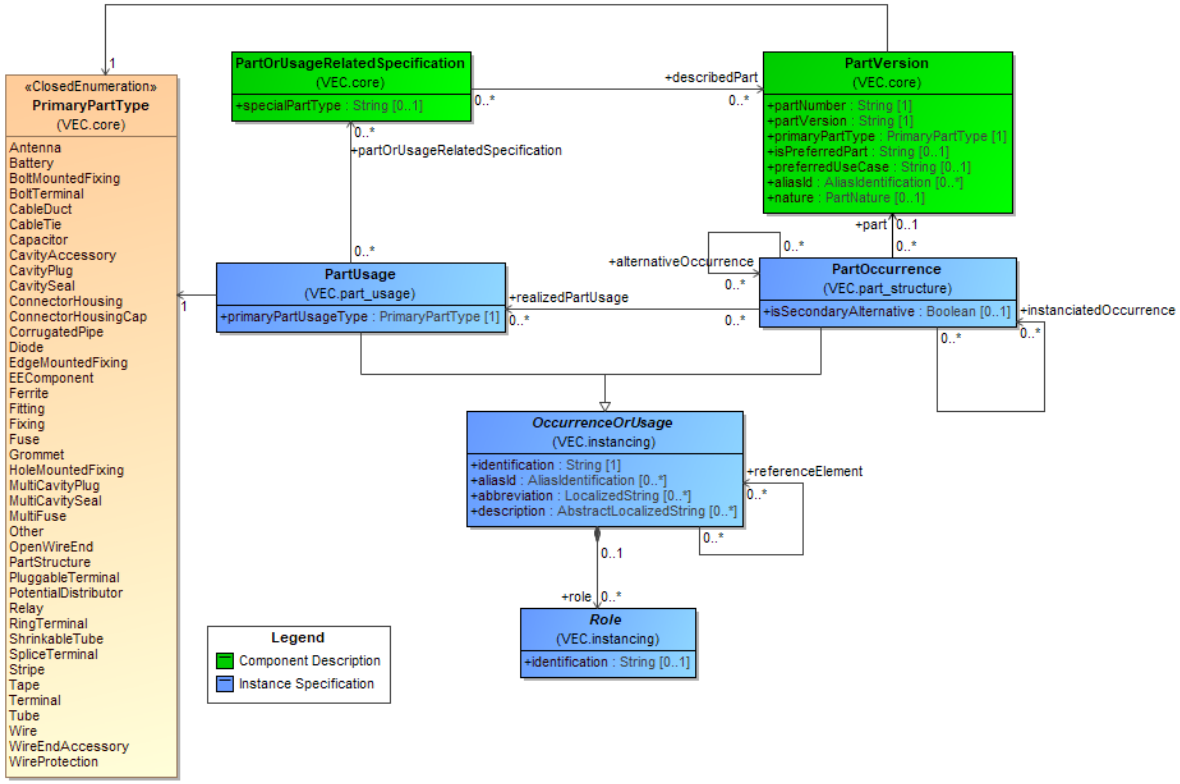


Figure 39: Instantiation of Components

It is a quite common pattern for complex product descriptions, to differentiate between types and their instances. A type is a general description that applies to all instances of this type. In the VEC the types of components are described with *PartVersions* and *PartOrUsageRelatedSpecifications*. They are describing the general properties of a certain component type (e.g. a connector housing with a certain *partNumber*). In order to define a more complex product with these elements, instances of the components are necessary (e.g. the connector housing for the left front light). To create such instances of components, the VEC provides two concepts: *PartOccurrence* and *PartUsage*.

The two concepts are necessary to fulfil the requirement for the VEC, that equal design elements are represented equally in the model, regardless of the level abstraction. This means for example, that the specification of a contacting must be represented always in the same way, regardless of the context either in an electrological specification or in a finalised product specification. The difference is that in an electrological specification in many cases defines only requirements for a component (e.g. the wire color, cross section area or number of cavities). In a finalized product specification, all components normally have assigned part numbers.

A *PartOccurrence* is a concrete occurrence of a certain *PartVersion* in a real product (Metaphorically spoken: if the product is broken into pieces, there will be a good chance

that one of the pieces is the *PartOccurrence* one is searching for). Therefore, the *PartOccurrence* references directly a *PartVersion*.

PartUsages shall be used for the specification of the elements on a more abstract level. This is the case either when not all technical properties required to select a correct *PartVersion* are known (e.g. only the cross section area of a wire is known, but not the color) or when not all usage properties are known that are required to create a *PartOccurrence.* This is true for example for a connectivity description and for a pure geometry description. The final occurrences come first into being when a connectivity description and a geometry description is combined. This is even true in the case of connectivity description where the part numbers for the wiring connections are already defined. The reason is that the wire lengths are still undefined, because the length information is dependent on a certain geometry. Additionally, it is possible that the same wiring definition is instantiated within different geometries, which might result in different *PartOccurrences*. For reasons of traceability, each *PartOccurrence* can reference back to the *PartUsage* it realizes.

In order to describe technical properties for a *PartUsage* without requiring a concrete *PartVersion* or a pseudo *PartVersion* the *PartUsage* can reference several *PartOrUsageRelatedSpecifications* like the *PartVersion.* As described in the chapter "part master data" the VEC allows the description of components through the combination of different characteristics (the *PartOrUsageRelatedSpecifications).* Therefore, a *PartUsage* can reference multiple *PartOrUsageRelatedSpecification*. Its primary characteristic is defined with the *PrimaryPartType*.

However different characteristics of component types may require different properties for instances of them. Therefore, a symmetrical concept to the *PartOrUsageRelatedSpecification* for instances was created in the VEC: The *Role.* Quite simplified: A *Role* is an instance of a *PartOrUsageRelatedSpecification* in the context of an *OccurrenceOrUsage* which is an instance of an abstract or concrete part (difference between *PartUsage* and *PartOccurrence*).

Roles, their subclasses and the sub elements of them are used whenever type specific usage information is required (e.g. a reference to a cavity in a certain use of a connector). Every other concept in VEC that requires references to instances uses these elements (e.g. the contacting). Since a *Role* can be used without knowing if it belongs to a *PartUsage* or a *PartOccurrence* concepts like the contacting can be reused regardless of the level of abstract (e.g. in a system wiring without concrete part numbers or in the product specification of wiring harness).

5.7.2 Instances of Wires

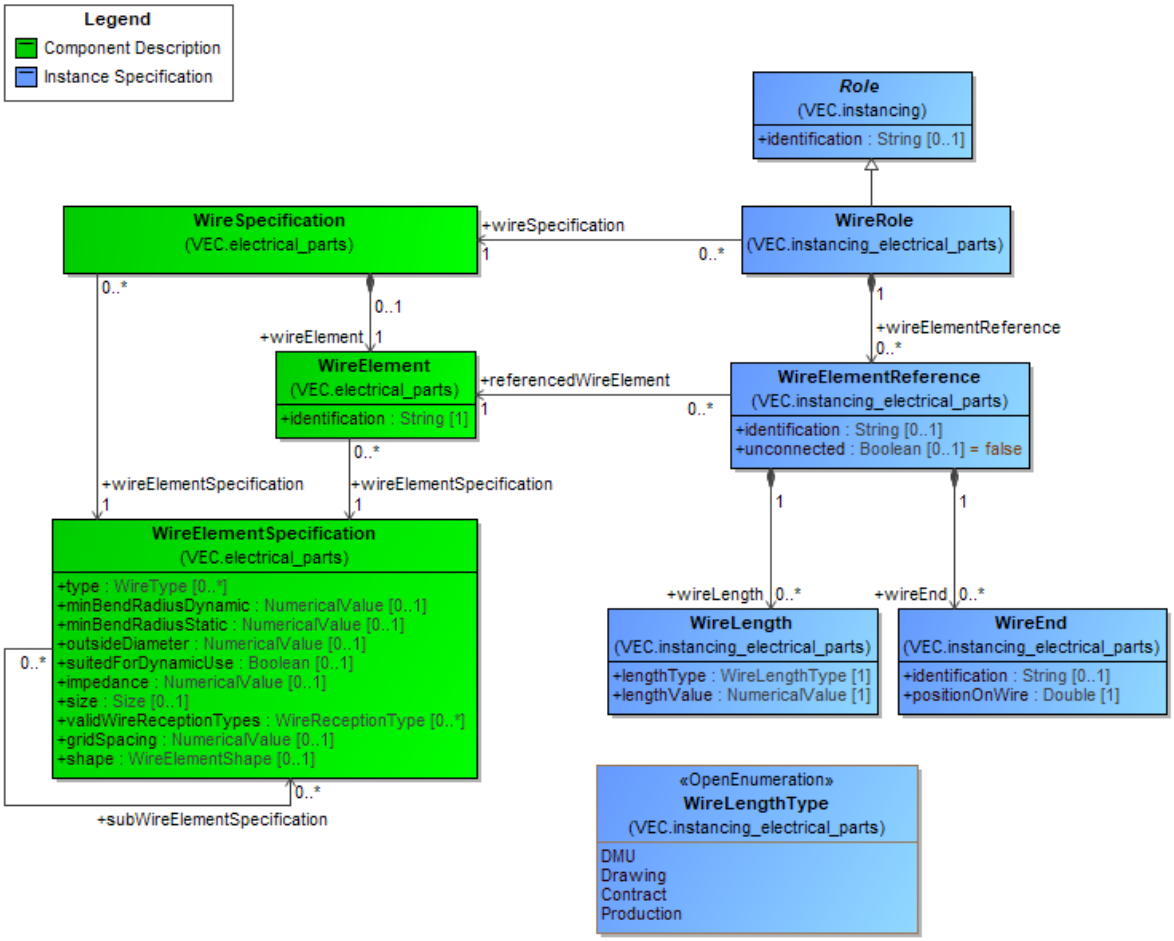


Figure 40: Instances of Wires

The diagram shows the mapping between the part master data of a wire (partly displayed on the left side) and the instance specific information (displayed on the right side).

5.7.3 Instances of Terminals

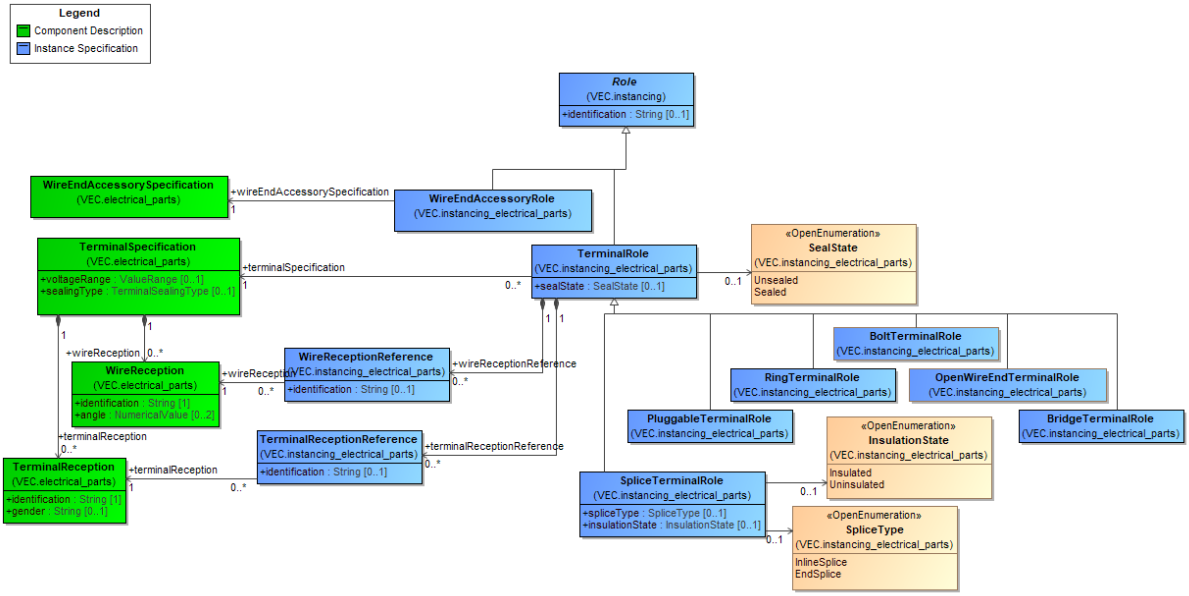


Figure 41: Instances of Terminals

The diagram shows the mapping between the part master data of a terminal (partly displayed on the left side) and the instance specific information (displayed on the right side).

5.7.4 Instances of Connector Housings

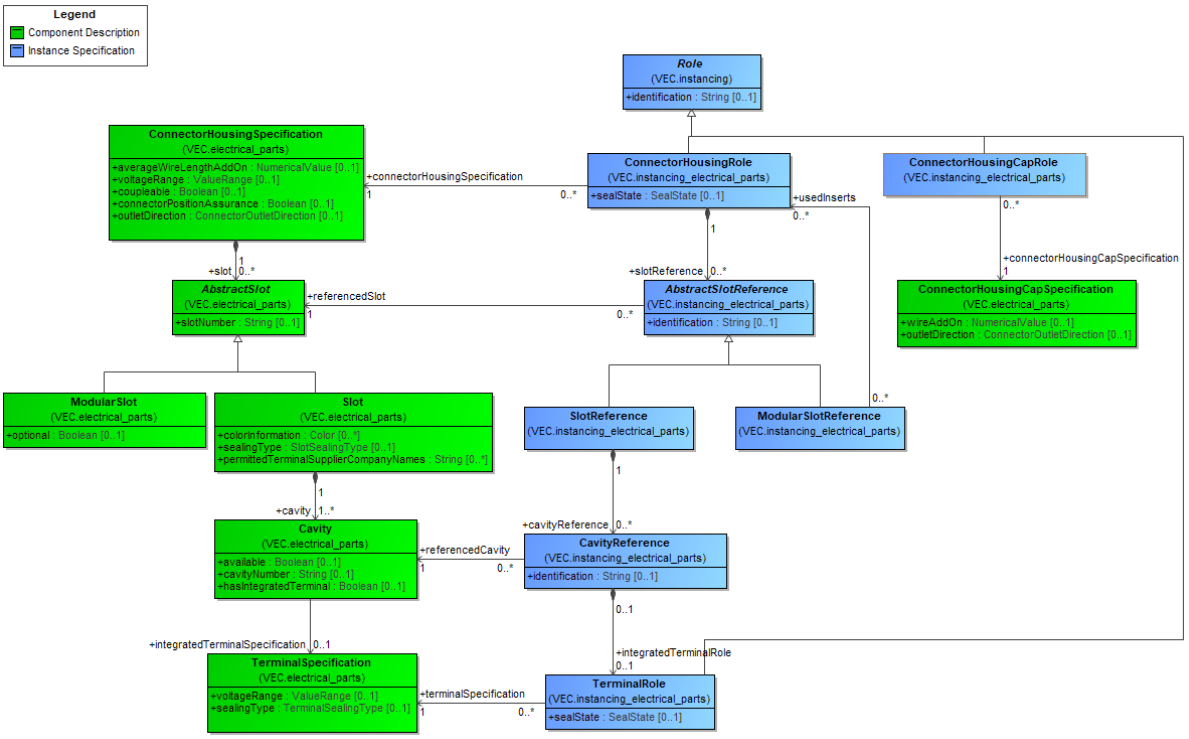


Figure 42: Instances of Connector Housings

The diagram shows the mapping between the part master data of a connector housing (partly displayed on the left side) and the instance specific information (displayed on the right side).

5.7.5 Instances of Cavity Seals and Cavity Plugs

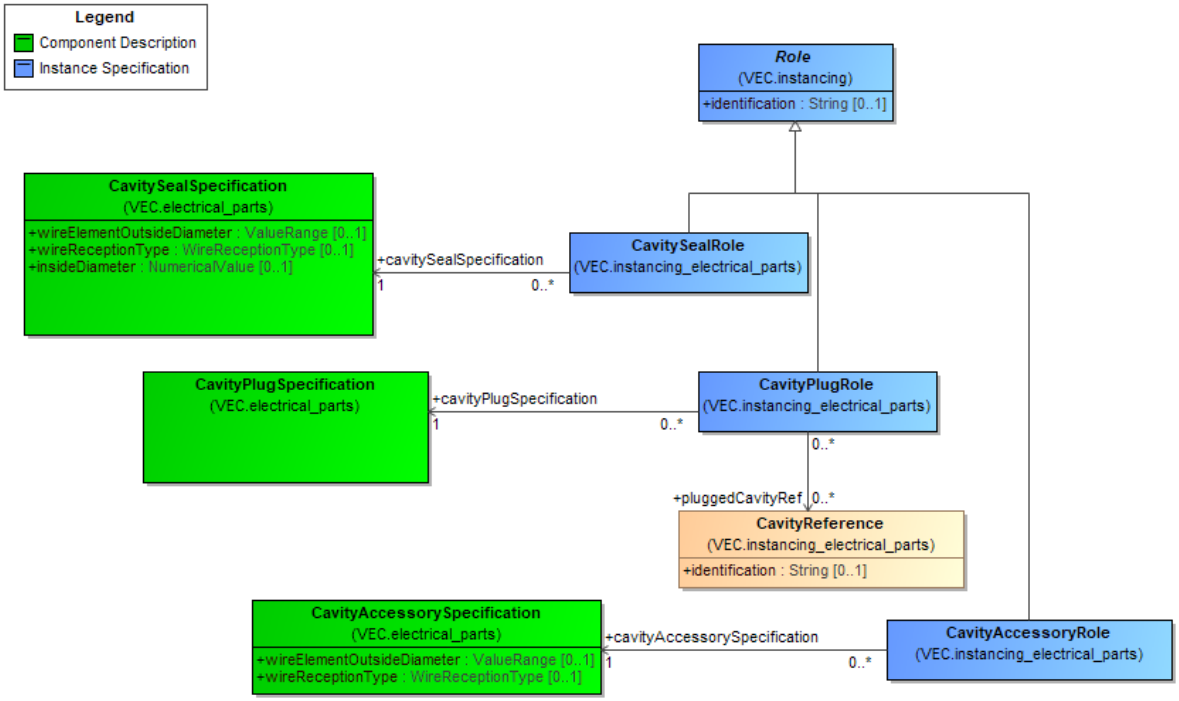


Figure 43: Instances of Cavity Seals and Cavity Plugs

The diagram shows the mapping between the part master data of a cavity seals and cavity plugs (partly displayed on the left side) and the instance specific information (displayed on the right side).

5.7.6 Instances of EE-Components

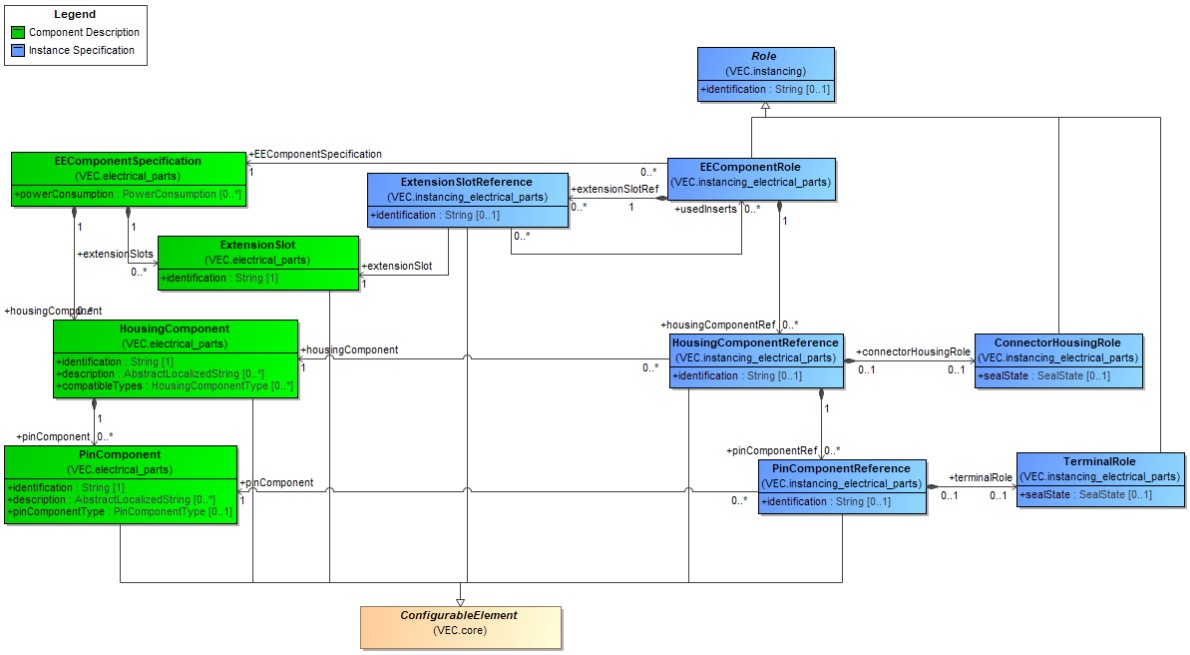


Figure 44: Instances of EE-Components

The diagram shows the mapping between the part master data of an EE-component (partly displayed on the left side) and the instance specific information (displayed on

the right side). Additionally, an *EEComponentRole* can reference one or more *ConnectorHousingRoles*, *AbstractSlotRoles* or *CavityReferences* onto which it is mounted.

5.7.7 Instances of Wire Protections

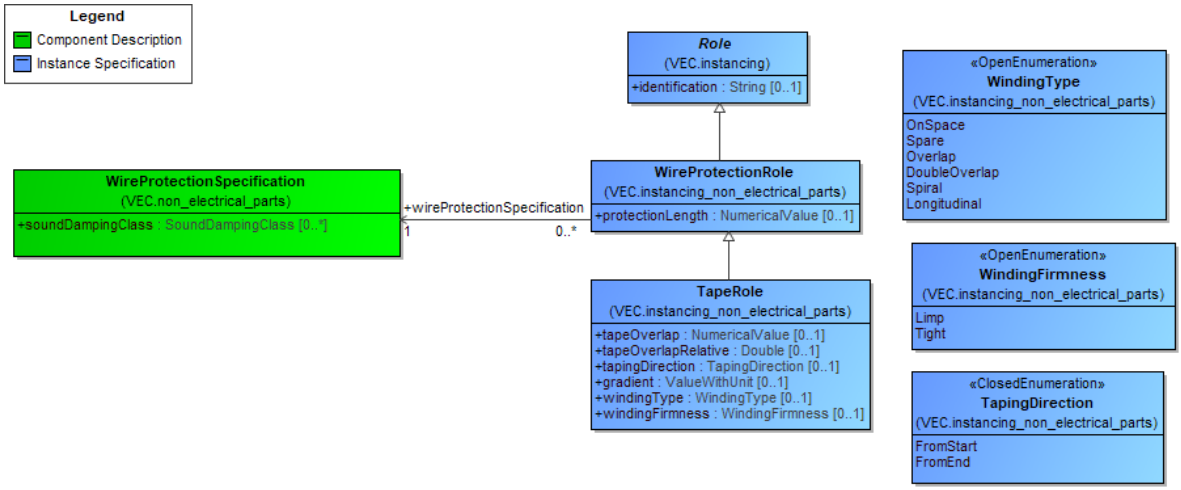


Figure 45: Instances of Wire Protections

The diagram shows the mapping between the part master data of a wire protection (partly displayed on the left side) and the instance specific information (displayed on the right side).

5.7.8 Instances of Fixings, Cable Ducts, Cable Ties and Similar

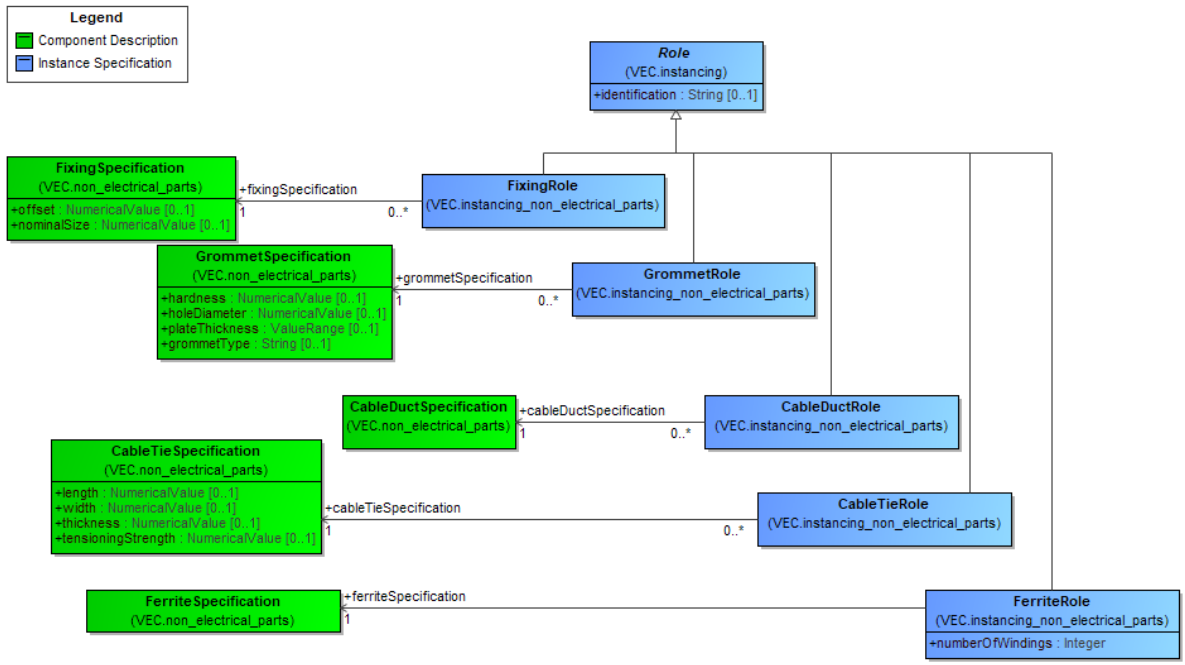


Figure 46: Instances of Fixings, Cable Ducts, Cable Ties and Similar

The diagram shows the mapping between the part master data of fixings and cable ducts (partly displayed on the left side) and the instance specific information (displayed on the right side).

5.7.9 Instances of Grommets

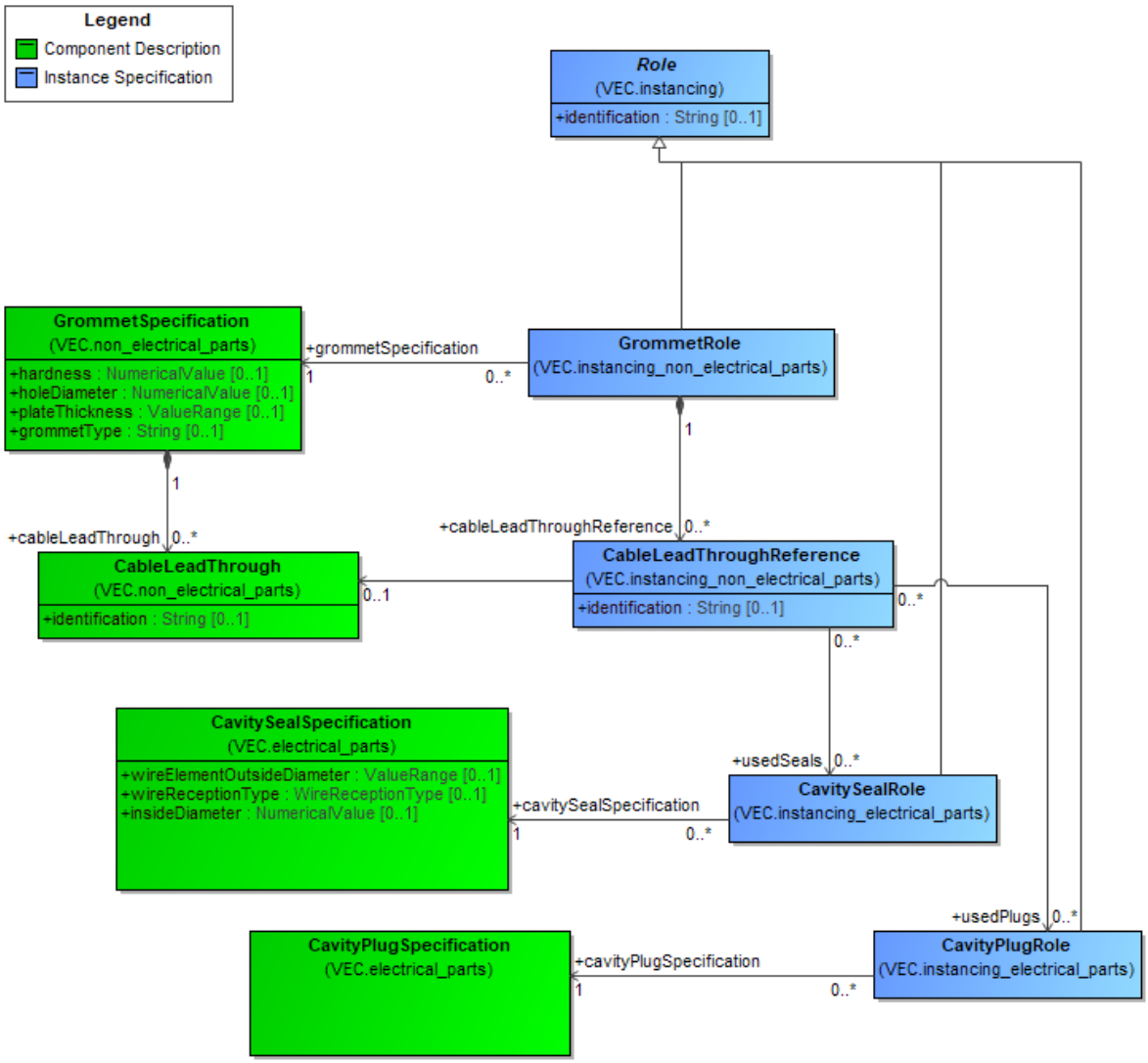


Figure 47: Instances of Grommets

5.7.10 Instances of undefined Components

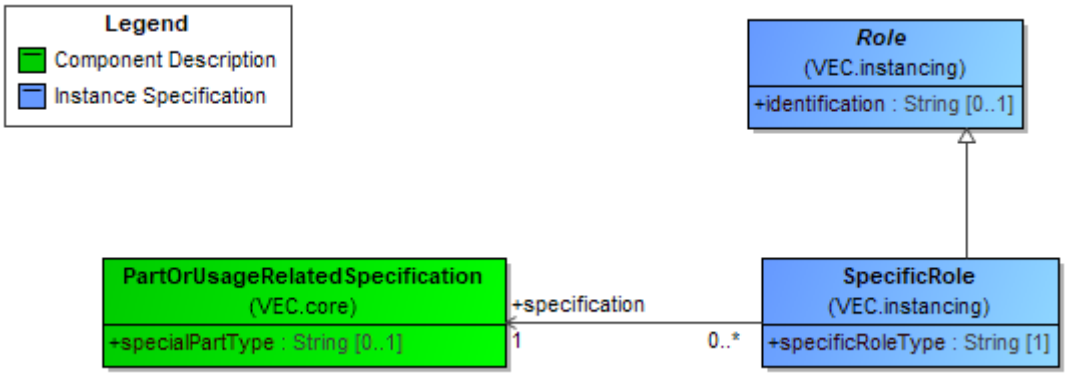


Figure 48: Instances of undefined Components

In some cases, it is necessary to specify and to use a component which is not defined in detail by the VEC. The recommended approach in such a case is to use a

PartOrUsageRelatedSpecification with *CustomProperties* to specify such a component. When this component must be instanced, a *SpecificRole* shall be used.

5.7.11 Instances of Placeable Components

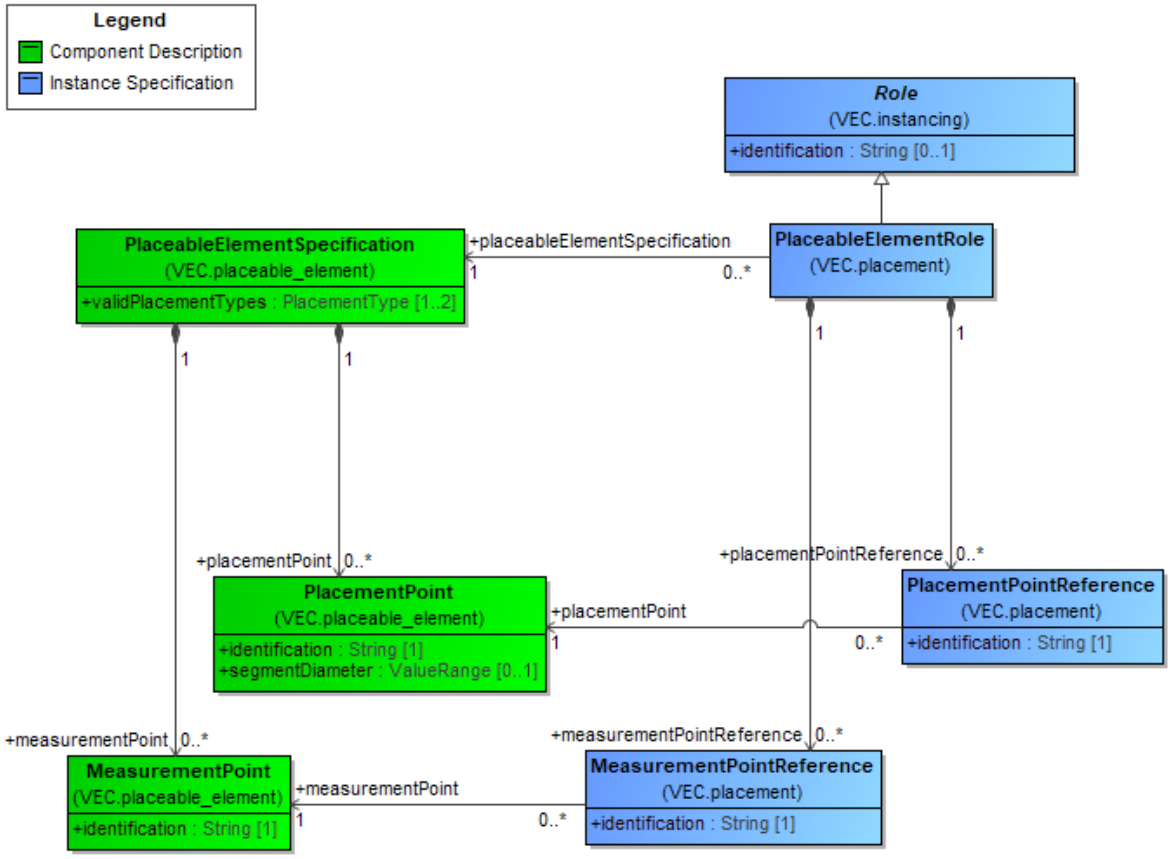


Figure 49: Instances of Placeable Components

The diagram shows the mapping between the part master data of a *PlaceableElement* (partly displayed on the left side) and the corresponding instance specific information (displayed on the right side).

5.7.12 Installation Instructions

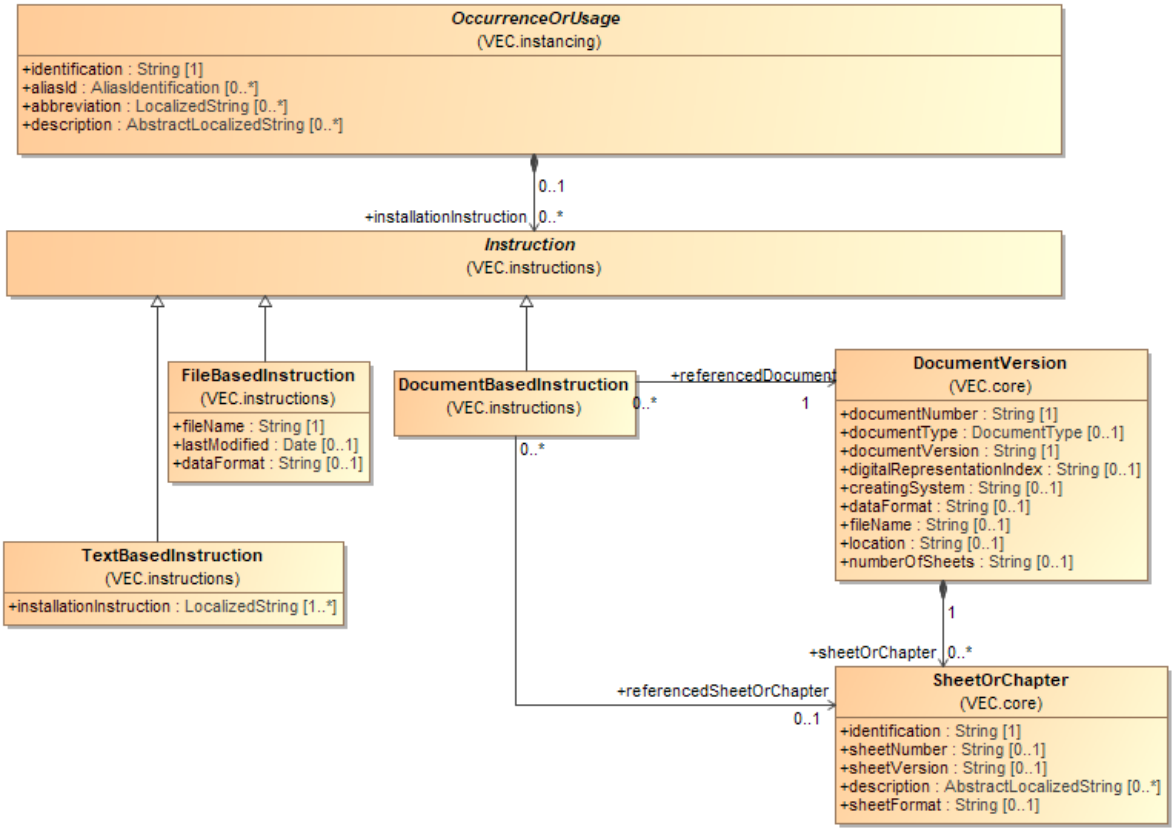


Figure 50: Installation Instructions

Basis for the definition of installation instructions is the abstract class *Instruction* which allows instructions to be specified as *TextBasedInstructions*, *FileBasedInstructions* or *DocumentBasedInstructions.*

Note: This diagram only shows the *Instruction* concept in the context of *OccurrenceOrUsage*. However, the *Instruction* concept is used in the context of *PartVersions* and several elements of a *ConnectionSpecification* as well.

Each *OccurrenceOrUsage* can define various *TextBasedInstructions* in order to exchange (normally) human readable installation instruction information.

Note: Unlike the KBL a *TextBasedInstruction* does not contain a type attribute. This is because a *TextBasedInstruction* is explicitly not wanted to be used as an extension mechanism. The declared extension concept of the VEC is the *CustomProperty* concept.

In addition, each *OccurrenceOrUsage* can define various *FileBasedInstructions*. This may be useful to reference for example files that contain graphical information e.g. in form of SVG.

In addition, each *OccurrenceOrUsage* can define various *DocumentBasedInstructions*. This allows to reference a *DocumentVersion* or alternatively a certain *SheetOrChapter* in a *DocumentVersion* that contains the instruction.

5.8 Composite Part Descriptions

The KBL defined a fixed hierarchy for composite parts, where different concepts covered specific aspects. These concepts were all represented in the model individually: *Harness*, *Module, HarnessConfiguration, Assembly.* From a structural point of view, all of them where quite the same, a composition of components (parts), the only real difference is the handling in process. However, the point of view might change the handling in the process.

For example, a module is assembled of components like connector housings, wires and terminals or pre-assembled parts (assembly) that are again assembled of connector housings, wires and terminals. From an abstract point of view a part regarded as a module by one process partner could be regarded as an assembly by another process partner and vice versa.

The VEC harmonizes these different concepts into one general concept for all these aspects. The following sections are describing this concept.

5.8.1 Multi-level Part Structure

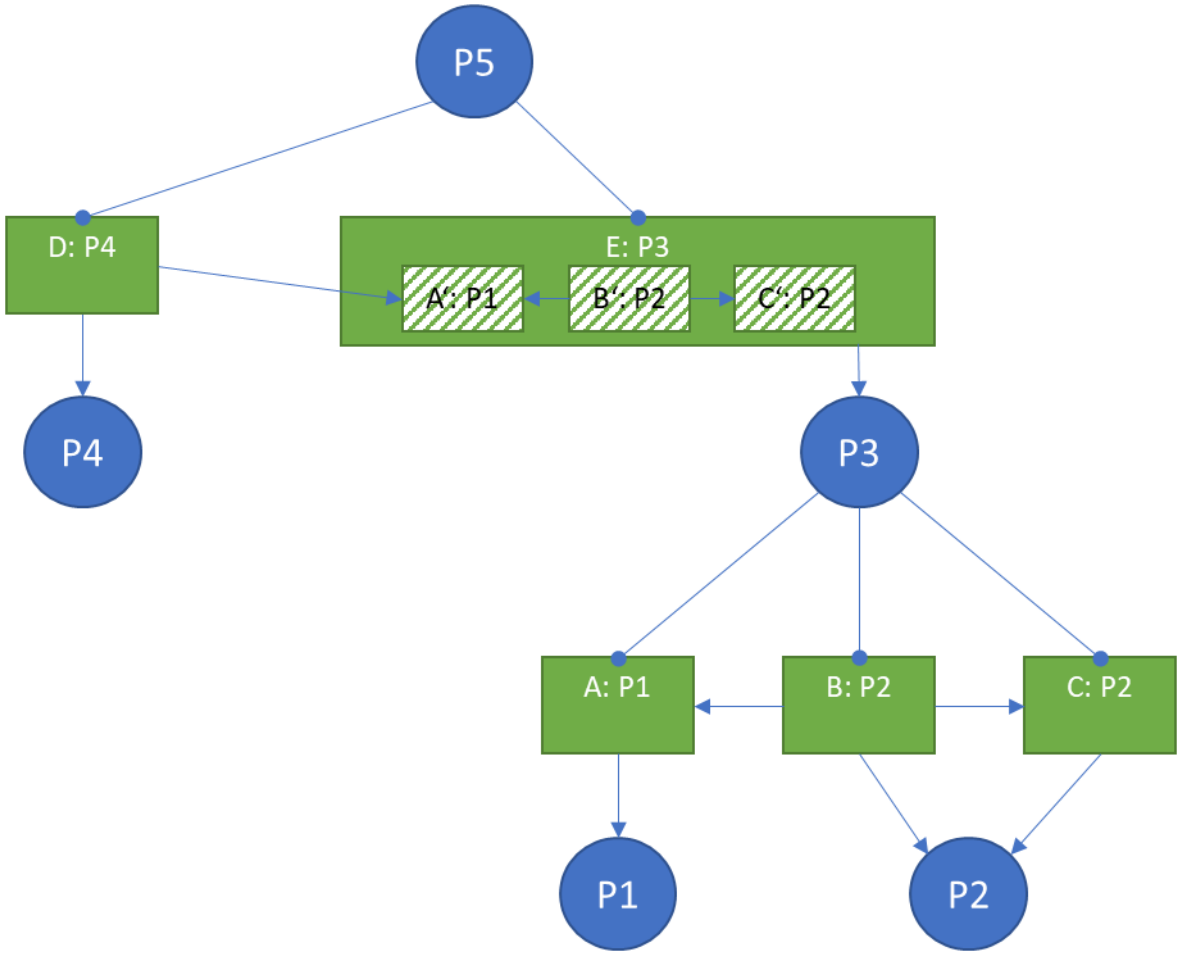


Figure 51: Multi-level Part Structure

The basic idea is the same as for *Multi-level bill of materials (BOM)*, where a product is defined in a parent-child, top-down method. It lists all raw materials, semi-finished

goods and sub-assemblies that are required to build the product in a hierarchical way. A component in such bill of material can consist of child components, which in-turn can have their own child components, and so on.

However, a regular bill of material just defines quantities of parts required (e.g. 4x P1, 1x P2, ...). Since the VEC not only describes the quantities of materials required to build a product, but also the way how the materials should interact (e.g. a specific terminal on a wire end in a defined cavity), the VEC requires an extended approach.

The diagram above illustrates an example for this and will be explained from bottom to top. P1 and P2 are basic components (parts with a part number). P1 could be a wire, P2 a terminal. To define some sort of composite component, the occurrences A, B and C are necessary (VEC class *OccurrenceOrUsage*). Those are required to specify the details of the composition with relationships among the occurrences and usage specific properties (e.g. wire length and contacting relationships). A composite component P3 is defined as consisting of occurrences A, B, and C.

The composite component P3 is used together with P4 to build P5. Again, occurrences are required. However, a single occurrence E for the composite part P3 is not sufficient, to specify the detailed composition of P5. Individual occurrences A', B' and C' of the sub-components of E are required for this. For example, to correctly specify the usage of a pre-assembled cable in a harness (a wire with connectors already attached to it), references to the sub-components are necessary (e.g. routing of the wire, position of connectors).

The *PartStructureSpecification* is used to define composite components based on occurrences (e.g. P3), the *PartWithSubComponentsRole* is used for the instancing of such composite parts (e.g. E: P3). For details see the next diagram.

5.8.2 Assemblies, Modules and Harness (Configurations)

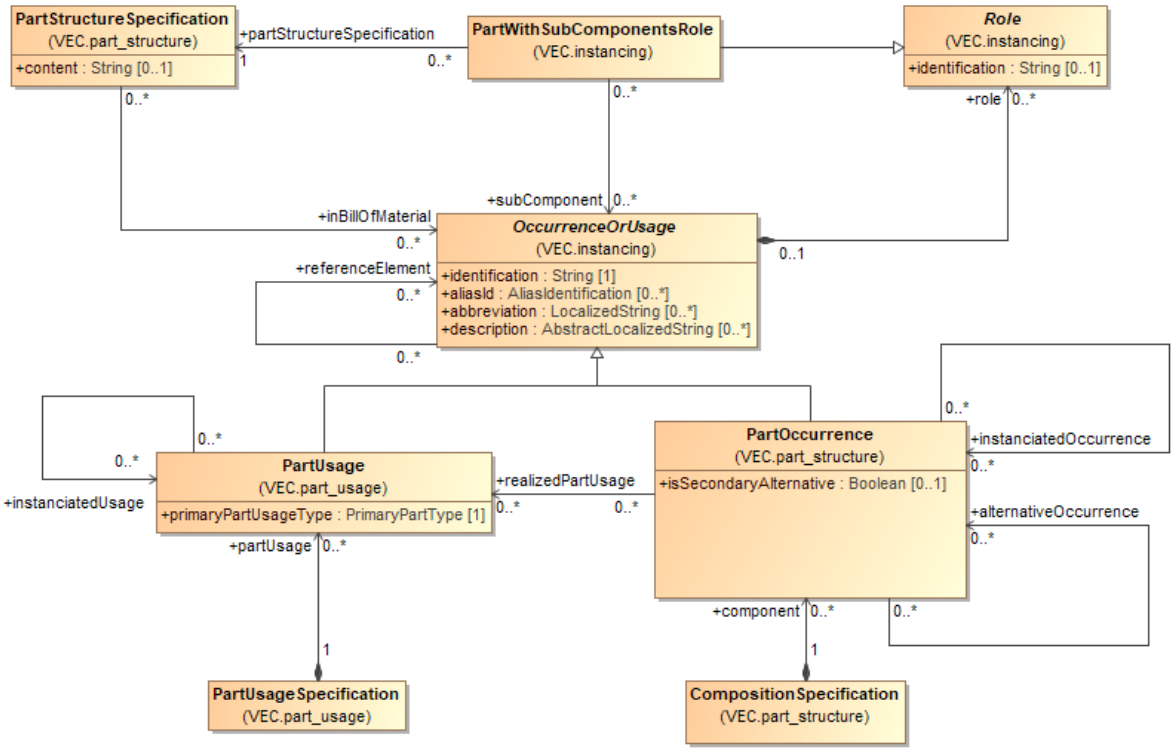


Figure 52: Assemblies, Modules and Harness (Configurations)

The VEC composite part concept has four key classes: the *PartStructureSpecification,* the *PartWithSubComponentsRole*, the *CompositionSpecification* and the *PartUsageSpecification*.

As mentioned in the example before, the *PartStructureSpecification* is used to define the bill of material for a composite part based on occurrences. Specific occurrences are used in favour of just referencing parts, to define not only quantities of material, but also the details of the composition. The *PartWithSubComponentsRole* is the corresponding instance of such a composite part.

Composite parts consist of *OccurrenceOrUsages,* so they can contain *PartOccurrences* (instances of a specific *PartVersion*) as well as *PartUsages* (instances of components where a specific *PartVersion* is not yet known). From a pure bill of material point of view this seems counter intuitive, since all materials in a bill of material should be well defined. However, there are scenarios where this is necessary. For instance, if a pre-assembled part (e.g. a cable) shall be used in a certain context, it is necessary, as mentioned before, to be able to reference the subcomponents. But if the subcomponents are not known in detail (no specific part number) *PartOccurrences* could not be used. Therefore, it is necessary to allow *PartUsages* in a composite part.

The *OccurrenceOrUsages* used for a composite part are just referenced and not owned (contained) by the *PartStructureSpecification* and *PartWithSubComponentsRole*. As a result, they can be shared among them. This is

especially useful in scenarios where a 150% product specification is done. Different variants of the product can reuse the same *OccurrenceOrUsages* without the need to copy them. The available *OccurrenceOrUsages* are defined by a *CompositionSpecification* (in case of *PartOccurrences*) or by a *PartUsageSpecification* (in case of *PartUsages)*.

***Note**: A PartVersion without a separate PartStructureSpecification shall be regarded as one atomic part out of a bill of material perspective even if it is referenced by a DocumentVersion containing a CompositionSpecificiation with several occurrences.*

5.8.3 Instantiation Approaches

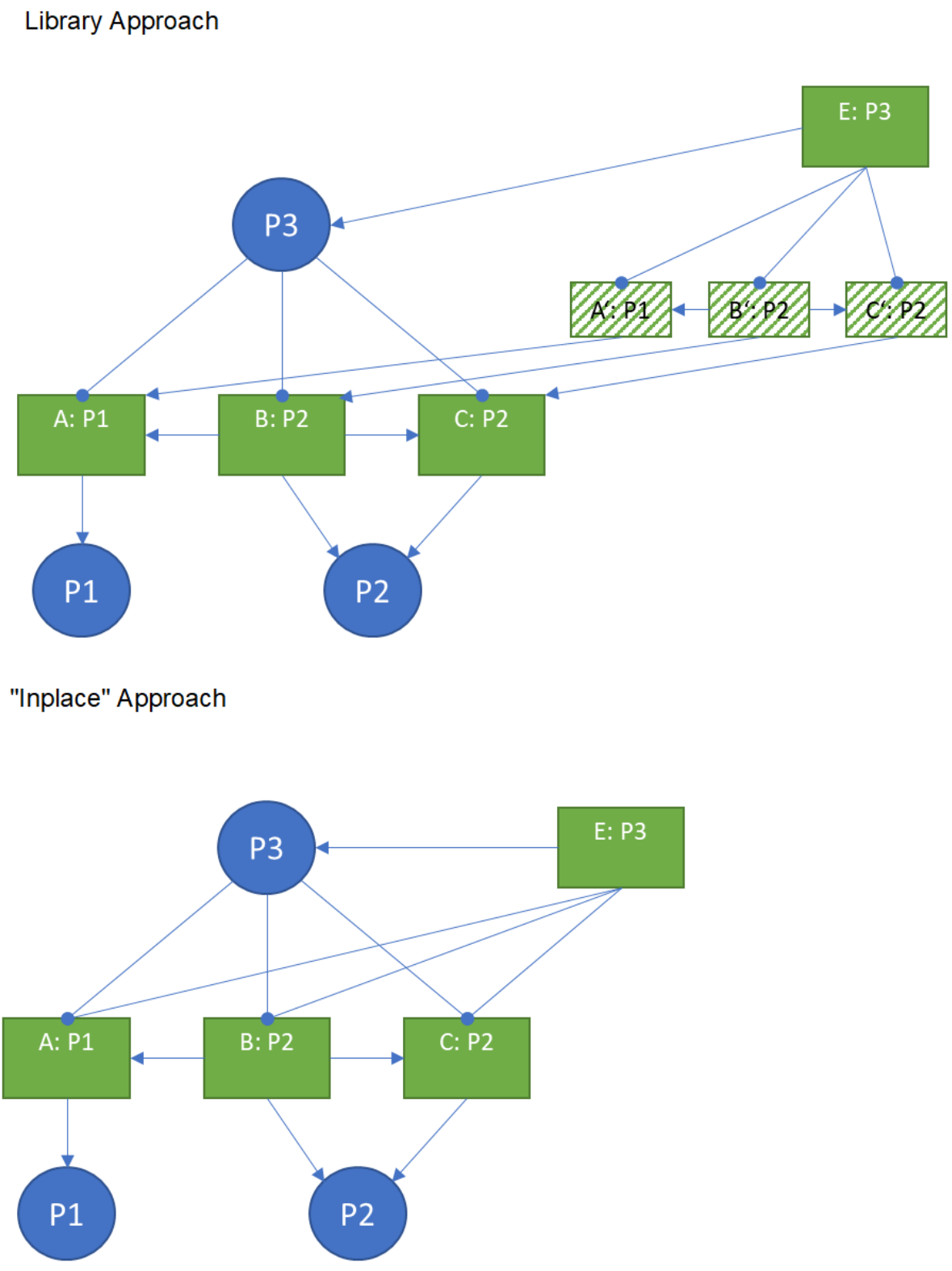


Figure 53: Instantiation Approaches

When a composite part is instantiated, an *OccurrenceOrUsage* with a *PartWithSubComponentsRole* is created and references the instantiated *OccurenceOrUsages* that belong to the composite part instance. The instantiated

occurrences are owned by a *PartUsageSpecification* / *CompositionSpecification* that is related to the context where the composite part is instantiated.

There are two approaches of how the definition and instantiation of composite parts is achieved: the library approach and the "in place" definition.

In the library approach, a composite part (e.g. a predefined USB cable) is defined in the component library and later used in different products. In this case, the part master definition of the composite part contains the *OccurrenceOrUsages* ("A: P1", "B: P2", "C: P2") that are required by the composite part and those are referenced by the *PartStructureSpecification* (P3). When the composite part is used (instantiated), the *OccurrenceOrUsages* from the part master definition are instantiated (copied) into the using context ("A': P1", "B': P2", "C': P2"). Those are then referenced by the *PartWithSubComponentsRole* (E: P3) and additional information can be attached to them. For traceability reasons, the *OccurrenceOrUsages* shall reference their origin in the part master definition via the instantiated*Usage/instantiatedOccurrence* association.

The "in place" definition is used for example for modules of a 150% harness. In this case, the composite part is not defined in a library, but in its only and single usage. In other words, the part master definition coincides with an instantiation of part. Therefore, the *PartStructureSpecification* (P3) and the *PartWithSubComponentsRole* (E: P3) reference the same set of *OccurrenceOrUsages* ("A: P1", "B: P2", "C: P2"). In this case both relationships of the *PartWithSubComponentsRole* and a *PartStructureSpecification* to the *OccurrenceOrUsage* (inBillOfMaterial & subComponents) shall be maintained consistently.

5.8.4 Harness and Variants

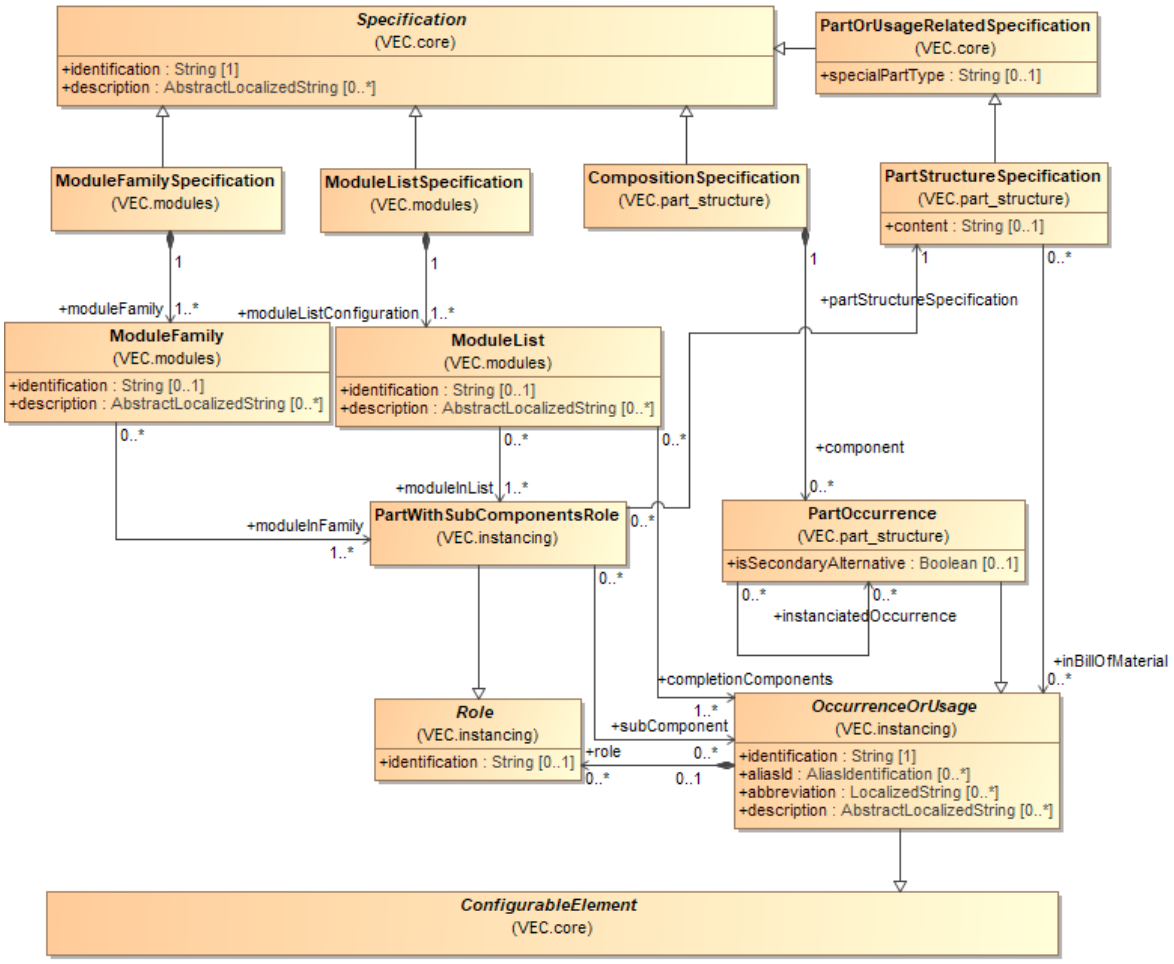


Figure 54: Harness and Variants

In addition to the *CompositionSpecification* and the *PartStructureSpecification*, the VEC defines two other concepts to control occurrences, especially composite parts (e.g. modules): *ModuleFamily* and *ModuleList*.

PartWithSubComponentsRoles referenced by a *ModuleFamily* respectively the *OccurrenceOrUsage* these *PartWithSubComponentsRoles* belong to are mutually exclusive.

A *ModuleList* specifies a set of *PartWithSubComponentsRoles.* The definition requires that if one or more of this *PartWithSubComponentsRoles* respectively the *OccurrenceOrUsage* these *PartWithSubComponentsRoles* belong to are part of an assembly the referenced *completionComponents* have to be added.

5.9 Topology and Geometry

This section covers the general definition of a topology, the definition geometric views of that topology, placement of *OccurrenceOrUsages* on a topology and other geometric aspects like dimensioning.

All geometric coordinate systems used in the VEC are right-handed (Right-hand rule). This also means, the direction of the z-Axis is in the direction of the cross product (X-Axis x Y-Axis).

5.9.1 Topology- and Topology Group Specification

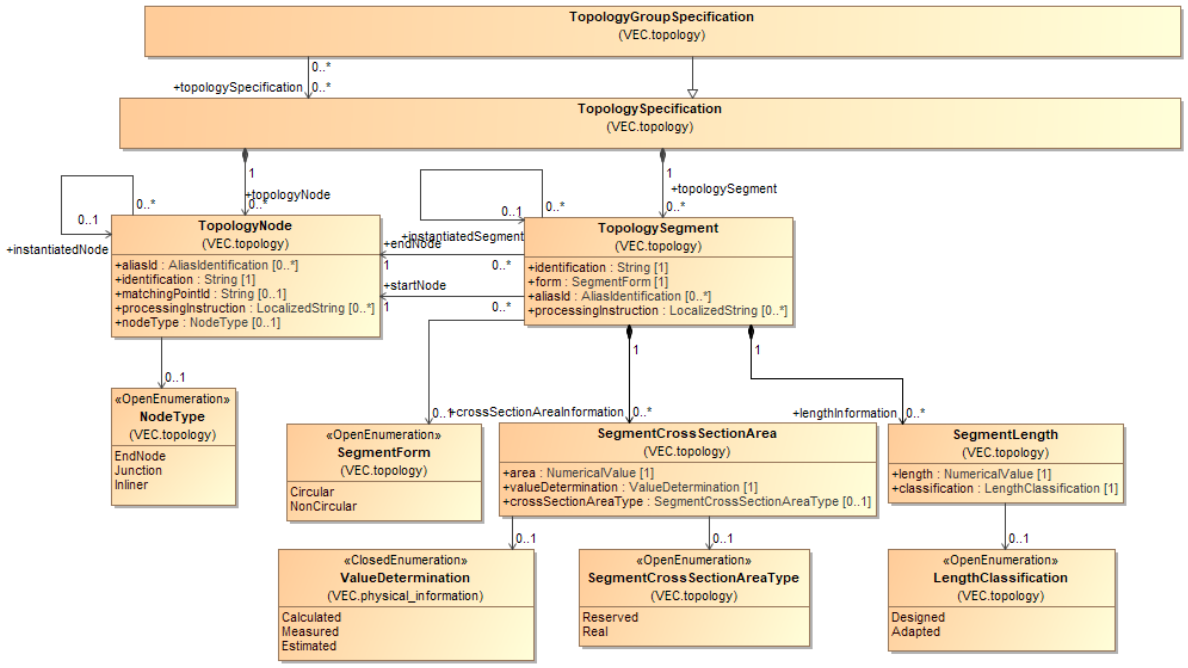


Figure 55: Topology- and Topology Group Specification

A *TopologySpecification* is a container for various *TopologyNodes*, *TopologySegments* and *Zones*. A *TopologyNode* represents an abstract place (meaning without concrete coordinates) in the wiring harness that can be typed either as *EndNode*, *JunctionPoint* or *Inliner*.

A *TopologySegment* represents a section of a wiring harness where no intermediate electrical contacts appear. In consequence, at the beginning and at the end of a *TopologySegment* the same wires go in and out. The beginning and the end are each defined by a dedicated reference to a *TopologyNode*. A *TopologySegment* can specify various *SegmentCrossSectionAreas*. These *SegmentCrossSectionAreas* are not supposed to express configuration-dependant values but different views (*Calculated*, *Measured* or *Estimated*) on the same type-dependant value whereupon the default-type is “reserved design space”. The same applies to *SegmentLength*.

In contrast to GEO 1.0 the VEC recognises an explicit grouping mechanism for topology. For this the VEC has introduced the class *TopologyGroupSpecification* which primary acts as a “normal” *TopologySpecification* being a container for various *TopologyNodes*, *TopologySegments* . However, a *TopologyGroupSpecification* can additionally reference the *TopologySpecifications* that it groups.

In some cases, a topology can be an instance of another topology. If a more complex assembly or small harness (e.g. like a door) is defined in some place (e.g. in a component library) and then reused elsewhere it might be necessary to instantiate the

topology in the target. To keep traceability in these cases, the *TopologySegment* and the *TopologyNode* have an *instantiated...* association to their source element.

5.9.2 Topology Zones

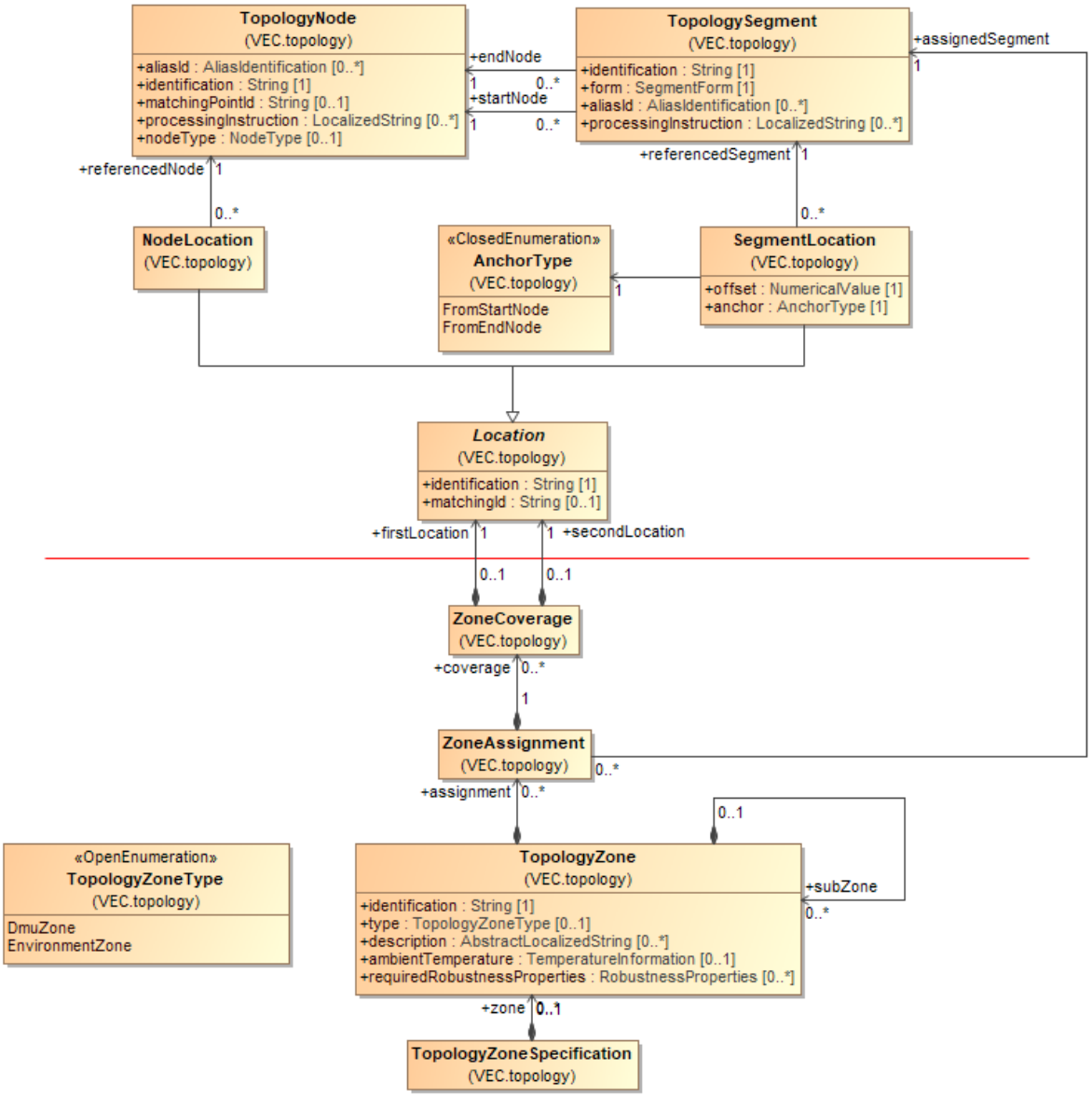


Figure 56: Topology Zones

A *TopologyZoneSpecification* is a container to define *TopologyZones.* A *TopologyZone* is a way to define areas on that share a set of properties (e.g. environmental influence).

To define the *TopologyZone* unambiguously an assignment of the topology elements to their zones is necessary. *TopologyNodes* are only connecting points for *TopologySegments* or for the placement of components and do not have any own extent in space. Therefore, *ZoneAssignments* are only possible for *TopologySegments*.

For more details, see the description of the relevant classes.

5.9.3 Hierarchical Topologies

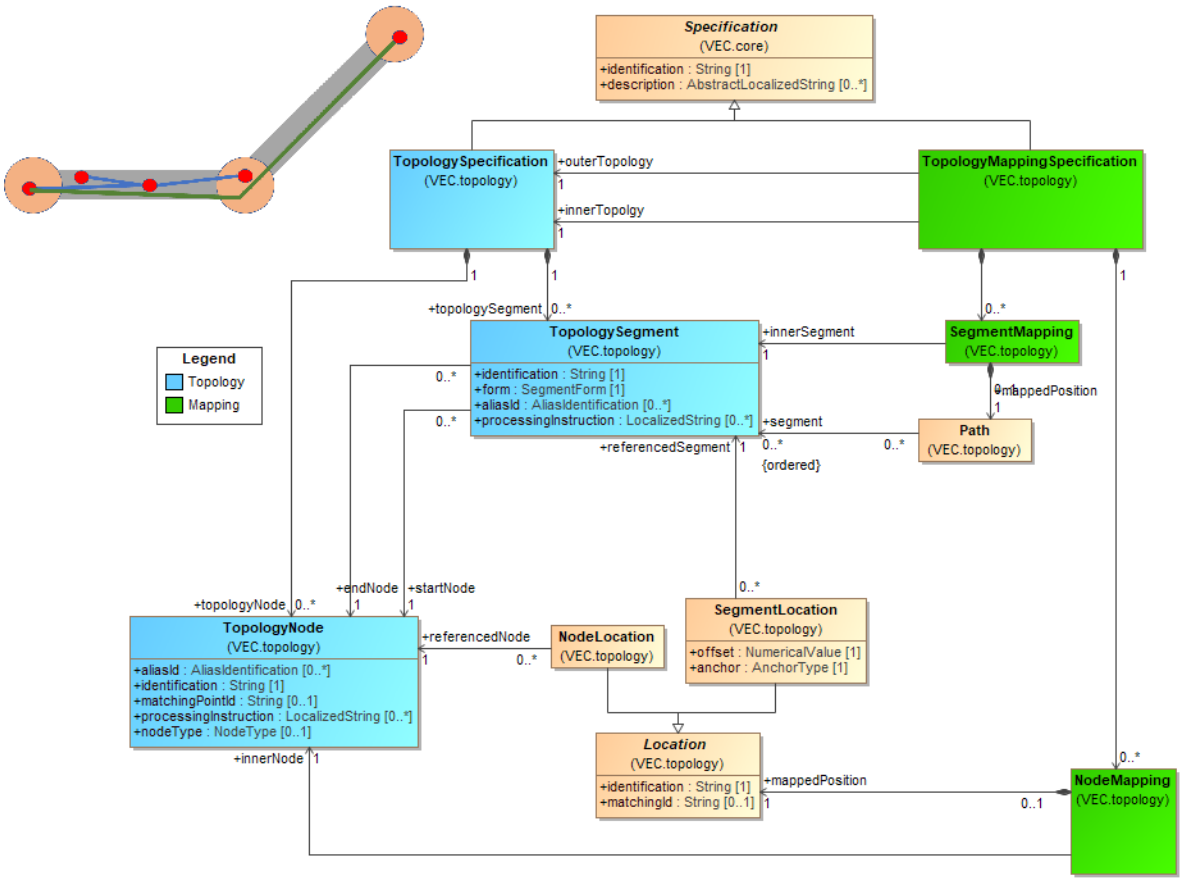


Figure 57: Hierarchical Topologies

Hierarchical topologies are a versatile concept to create harness topologies that have an advanced structural design. For historical reasons, the semantics defined by the VEC for topologies are quite strict and simple, yet sufficient for most cases of traditional wiring harnesses. However, with the advent of modern wiring harness requirements, more advanced concepts are needed for certain applications.

In a regular topology, all wires are routed equally "inside" through a *TopologySegment.* All wires end at a *TopologyNode,* there are no connectors or splices positioned within a *TopologySegment.* All protections, fixings, tapings etc. are placed "around" the wires in a definable order.

With the integration of new technologies (e.g. high voltage, high data rate bus systems) and the improvements of design processes (e.g. integrated 3D design process with traceability and round-trip capabilities) more complex designs must be expressed in a concise, digital evaluable way. Examples for such use cases are:

- Structured harness bundles (e.g. specific wires in a bundle taped together).
- Segments of splices folded into a main bundle in a specific way.
- Reuse of shared parts (e.g. assemblies, right / left back door)
- Refinement of a topology in the design process (e.g. splice positioning after the "geometry design".

To address these requirements, the VEC introduced the concept of hierarchical topologies. In the top left corner of the diagram a simplified illustration shall serve as small example to make the concept more understandable.

A *TopologyMappingSpecification* is used to associate an "inner" topology with an outer topology. In the illustration the "outer" topology is displayed with grey segments and orange nodes. The two "inner" topologies are displayed with blue & green segments and red nodes.

A *NodeMapping* is used to define how the nodes of the inner topology relate to the outer topology. An inner node can be either placed exactly on / in an outer node or somewhere within a segment. For the definition of the position of the inner node, the concept of locations is reused.

A *SegmentMapping* is used to define how the segments of the inner topology relate to the outer topology. An inner segment can run through multiple segments or to stay in a single one. In any case, all outer *TopologySegments* to which it relates are referenced by a *Path*.

Some additional restrictions apply on the mapping:

- The representation of the inner / outer relationship with a *TopologyMappingSpecification* allows the creation of n:m relationships between topologies. However, it is forbidden to create "circular" mappings. If n:m mappings are created, it must be ensured that those mappings shall not exist at same time, or in other words they must be the result of product variance or reuse in different variants.
- "Cross Topology assignment of Components" is forbidden. That means wires are routed strictly in the elements of a single *TopologySpecification* and other components are placed strictly on a single *TopologySpecification.*
- If a *TopologySpecification* is mapped as an inner topology, all its elements shall be mapped.
- A creating system is responsible that the constraints of the topologies are maintained (e.g. that length constraints between inner and outer topologies are satisfied.

Note: *This concept was introduced newly with VEC Version 1.2.0. As there are currently no known systems that implement such a concept further detailing of its usage will be done in the Implementation Guidelines accompanying implementations as they occur.*

5.9.4 Bending Restrictions

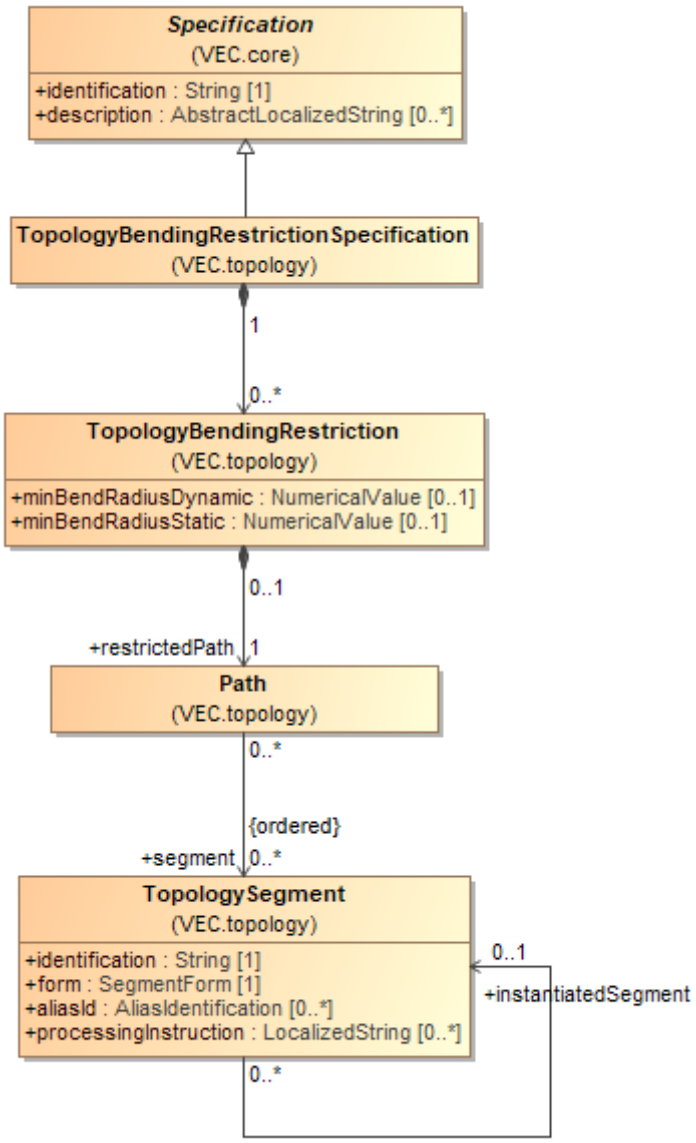


Figure 58: Bending Restrictions

A bending restriction defines restrictions on the bendability of a path in the topology. See *TopologyBendingRestriction* for a detailed description.

5.9.5 2D-Geometry

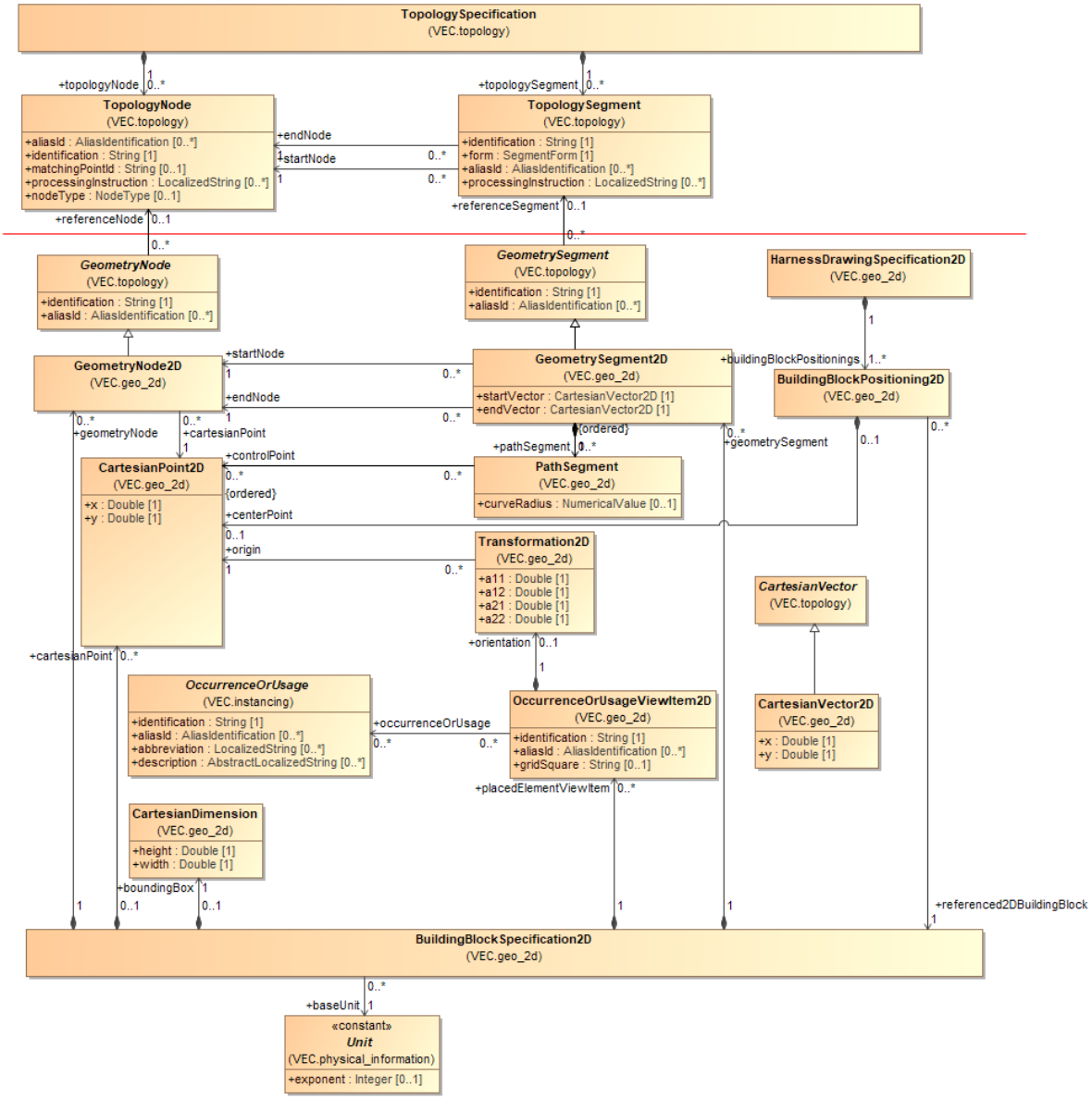


Figure 59: 2D-Geometry

A *HarnessDrawingSpecification2D* represents a specific harness drawing. Each *HarnessDrawingSpecification2D* is a container for various instances of the class *BuildingBlockPositioning2D*. Each *BuildingBlockPositioning2D* determines the positioning of a certain *BuildingBlockSpecification2D* in the context of the *HarnessDrawingSpecification2D* by referencing the relevant *BuildingBlockSpecification2D* and specifying the target *centerPoint* as *CartesianPoint2D*.

A *BuildingBlockSpecification2D* represents a specific subset of one or more harness drawings. Each *BulidingBlockSpecification2D* specifies its own *CartesianDimension*. Furthermore, each *BulidingBlockSpecification2D* is a container for various instances of the classes *GeometryNode2D*, *GeometrySegment2D*, *ContactPoint2D* and *OccurrenceOrUsageViewItem2D*.

Each *GeometryNode2D* references a *CartesianPoint2D* which determines its graphical position within the *BuildingBlockSpecification2D*. Moreover, it can specify which *TopologyNode* it represents.

Each *GeometrySegment2D* can specify an ordered set of *PathSegments* each defining a certain section of its graphical representation. Moreover, it can specify which *TopologySegment* it represents.

Note: The data model does not constrain a *GeometryNode2D* respectively a *GeometrySegment2D* to reference a *TopologyNode* respectively a *TopologySegment*. This allows a VEC e.g. to transport graphical 2D-information for its own. However, for the description of lengths and dimensions as well as segment cross-section-areas and placements the VEC requires adequate references to the relevant elements in the topology.

Each *OccurrenceOrUsageViewItem2D* can reference which *OccurrenceOrUsages* it represents. Furthermore, it can specify a *Transformation2D* which determines its graphical position and orientation within the *BuildingBlockSpecification2D.*

5.9.6 3D-Geometry

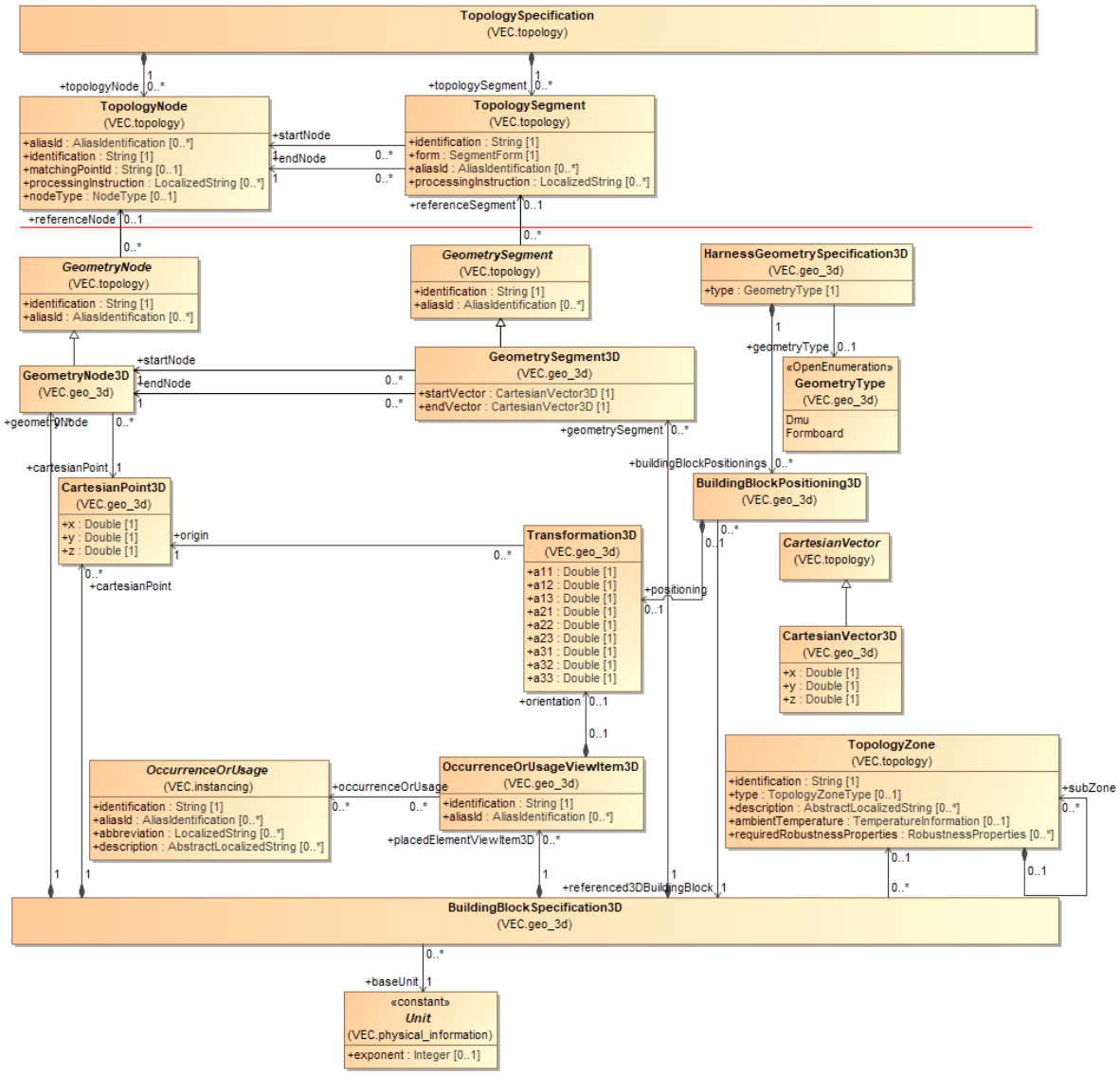


Figure 60: 3D-Geometry

A *HarnessGeometrySpecification3D* represents either a 3D-model in the car coordinate system or a 3D-model in the formboard system. Each *HarnessGeometrySpecification3D* is a container for various instances of the class *BuildingBlockPositioning3D*. Each *BuildingBlockPositioning3D* determines the positioning of a specific *BuildingBlockSpecification3D* in the context of the *HarnessGeometrySpecification3D* by referencing a *BuildingBlockSpecification3D* and optionally specifying a *Transformation3D* for the case that a coordinate transformation is required.

A *BuildingBlockSpecification3D* represents a specific geometrical subset normally belonging to a certain *Zone*. Each *BuildingBlockSpecification3D* is a container for various instances of the classes *GeometryNode3D*, *GeometrySegment3D*, *CartesianPoint3D* and *OccurrenceOrUsageViewItem3D*.

Each *GeometryNode3D* references a *CartesianPoint3D* which determines its position within the *BuildingBlockSpecification3D*. Moreover, it can specify which *TopologyNode* it represents.

Each *GeometrySegment3D* can specify an ordered set of *Curves* in order to approximate its geometry (explained in detail on the next diagram). Moreover, it can specify which *TopologySegment* it represents.

Note: The data model does not constrain a *GeometryNode3D* respectively a *GeometrySegment3D* to reference a *TopologyNode* respectively a *TopologySegment*. This allows a VEC e.g. to transport 3D-information independently from a *TopologySpecification*. However, for the description of dimensions including tolerances as well as segment cross-section-areas and placements especially *OnWayPlacements* the VEC requires adequate references to the relevant elements in the topology.

Each *OccurrenceOrUsageViewItem3D* can reference which *OccurrenceOrUsages* it represents. Furthermore, it can specify a *Transformation3D* which determines its position and orientation of the component's 3D model within the space of the *BuildingBlockSpecification3D.* If no *Transformation3D* is defined, the *OccurrenceOrUsageViewItem3D* is meant to be placed implicitly by its position in the topology (e.g. wire protections with an *OnWayPlacement*.

5.9.7 3D Curves

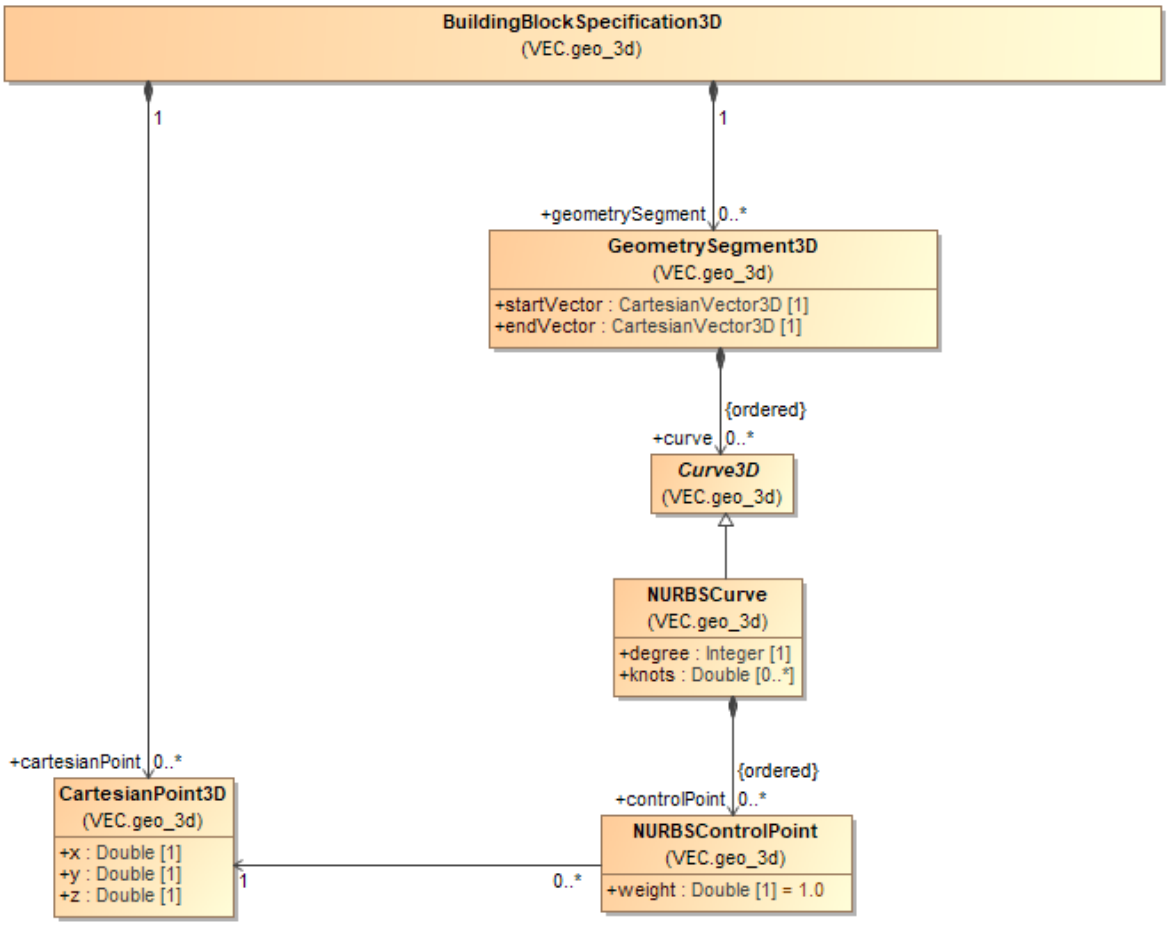


Figure 61: 3D Curves

The centre line of a *GeometrySegment3D* is defined by one or more *Curve3D*. This abstract class is designed to be an extension point for different curve implementations. At the moment the only curve implementation that is supported by the VEC are *NURBSCurves.* The details are described in the documentation of the individual classes.

5.9.8 Locations

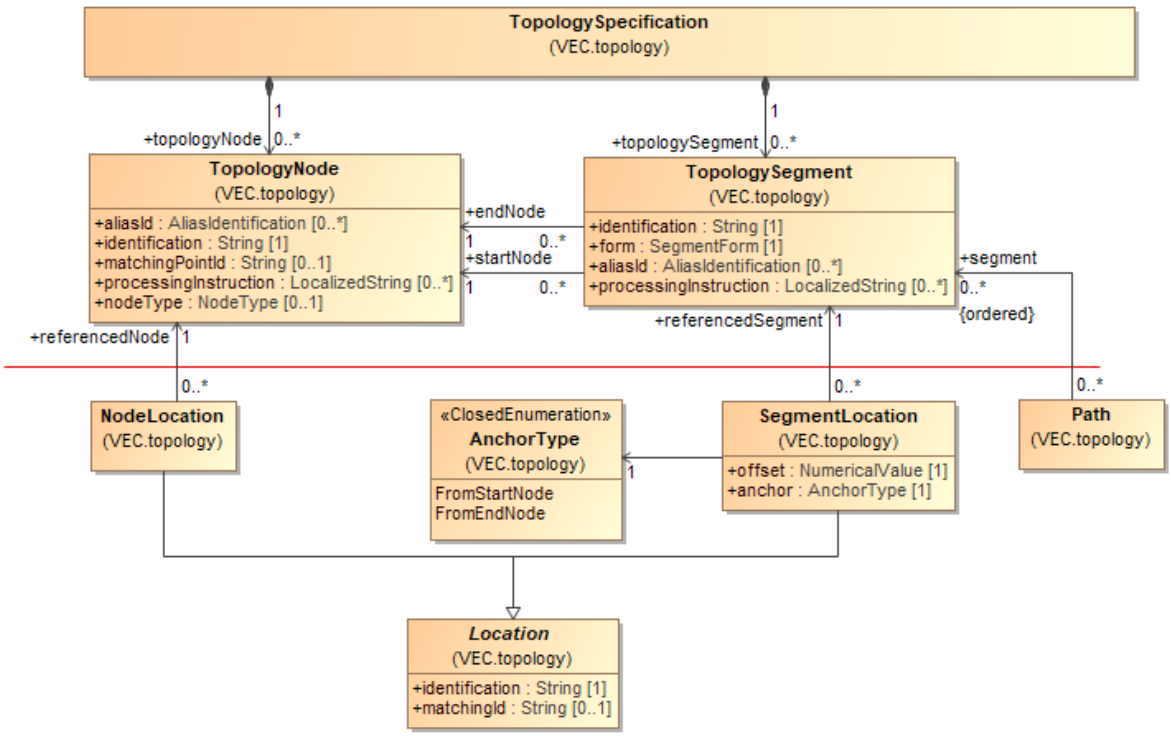


Figure 62: Locations

In order to describe a wiring harness, the abstract information of a topology must be enhanced with information about the concrete positions of components in the topology. For purposes of clarity the explanation of this concept is split into two diagrams. This diagram explains how *Locations* are defined onto the *Topology*. The next diagram shows how these locations can be used to define *Placements* of components and *Dimensions*.

The VEC defines two types of *Locations*: *NodeLocations* and *SegmentLocations*. A *NodeLocation* references one *TopologyNode*. A *SegmentLocation* defines a single point on a *TopologySegment* located between its *startNode* and *endNode*. To achieve this, a *SegmentLocation* references the relevant *TopologySegment* and specifies as anchor the start- or end node as well as the offset from that anchor.

Note: In contrast to KBL 2.3 and GEO the VEC locations on segments are expressed by anchor and offset and not through relative measurements. This enables ease of handling in case of segment-length changes.

5.9.9 Placement and Dimensions

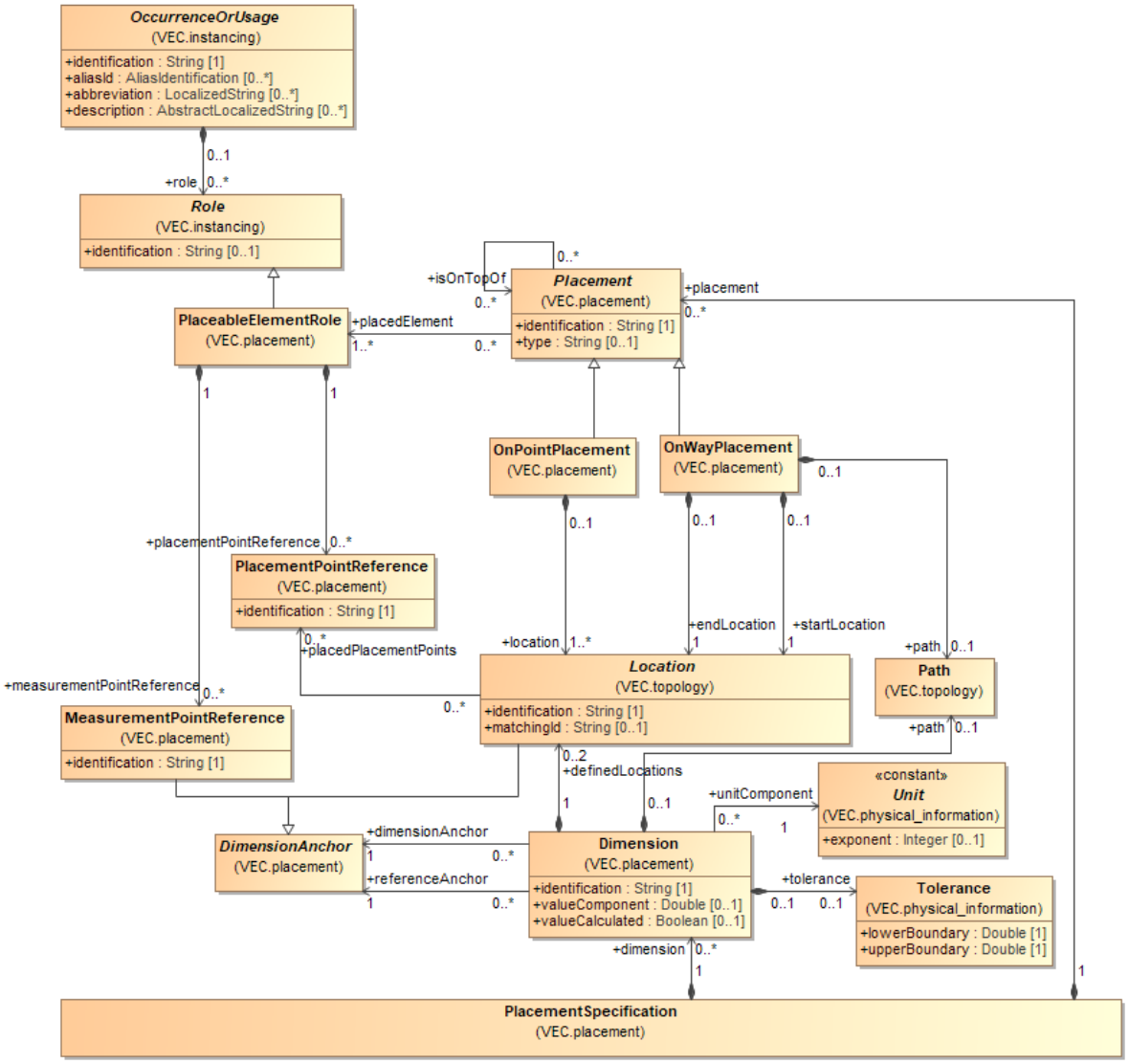


Figure 63: Placement and Dimensions

A *PlacementSpecification* is used to define the *Placement* of *OccurrenceOrUsages* in a topology as physical elements that build this topology and to define *Dimensions* between these elements. Each *PlacementSpecification* is a container for various *Placements* and *Dimensions*.

Each *OccurrenceOrUsage* that is placeable in the topology defines a *PlaceableElementRole.* A *PlaceableElementRole* can define various *PlacementPointReferences* and *MeasurementPointReferences*, which are instances of the corresponding elements *PlacementPoint* and *MeasurementPoint* defined in the part master data of the component. The position in the topology of a *PlaceableElementRole* is defined with a *Placement*. The multiplicity of this association is caused by the support of variance in the VEC. For a concrete configuration, there should be only one *Placement* for one *PlaceableElementRole.*

The VEC specifies two individual types of *Placements*: *OnPointPlacements* and *OnWayPlacements*.

Each *OnPointPlacement* defines one or more *Locations* in order to determine the positioning of the *placedElement*. More than one *Location* is necessary to specify the exact *Placement* of *placedElements* with more than one *PlacementPointReferences*, e.g. the different entries in case of a cable duct. Each *Location* can therefore reference a *PlacementPointReference.*

An *OnWayPlacement* references one *Location* as *startLocation* and another *Location* as *endLocation* in order to determine the positioning of *placedElements* like tapes that cover a way in the topology. The positioning can be described even more precise by using the *OnWayPlacement* with a *Path* through the topology.

Both *OnPointPlacements* and *OnWayPlacements* reference one or more *PlaceableElementRoles* as *placedElements*. The possibility to reference more than just one *PlaceableElementRole* permits to express that configuration dependant the one or other *PlaceableElementRole* is placed. Finally, both *OnPointPlacements* and *OnWayPlacements* can reference other *Placements* on which they are placed on top of in order to determine the arrangement of layering (e.g. a tape around a tube).

Note: When using the *isOnTopOf*-relationship all other underlying *Placements* shall be referenced on which the referencing *Placement* is on top of and not only the direct neighbour. The reason for this is that *Placements* are *ConfigurableElements* and so are potentially not present in all configurations.

Since the VEC can be used to exchange a product definition, it must be possible to define *Dimensions* with tolerances between the elements in addition to its exact positions in the topology.

Each *Dimension* references two *DimensionAnchor*, one as *referenceAnchor* and one as *dimensionAnchor* and defines the distance in between as *dimensionValue*, optionally together with a tolerance (included in the definition of *NumericalValue*). In case of ambiguity regarding the path between the two *Locations*, the *Dimension* shall specify the relevant *Path*.

A *DimensionAnchor* can be either a *MeasurementPointReference* or a *Location*. A *MeasurementPointReference* is used when the *Dimension* should reference an element on the component which is outside of the topology (e.g. a certain edge or a bolt). If a *Location* is referenced it can be either a *Location* defined by a *Placement* (if the dimension should reference the position of a component) or a *Location* defined by the *Dimension* itself if an element of the topology with no component on it must be referenced (e.g. a *TopologyNode*).

5.9.10 Default Dimensions

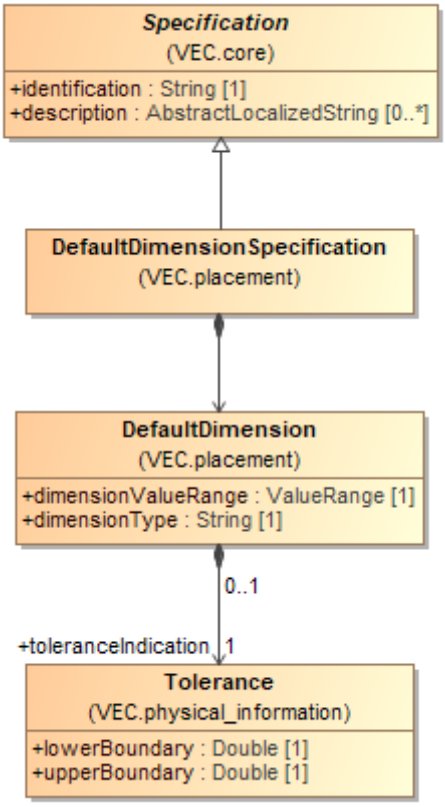


Figure 64: Default Dimensions

DefaultDimension can be used to define standard tolerances that apply to certain elements. Typically they are represented in a document or a drawing as a table (e.g. for wires with length < x; tolerance a, x < length < y; tolerance b and so on).

5.9.11 Routing

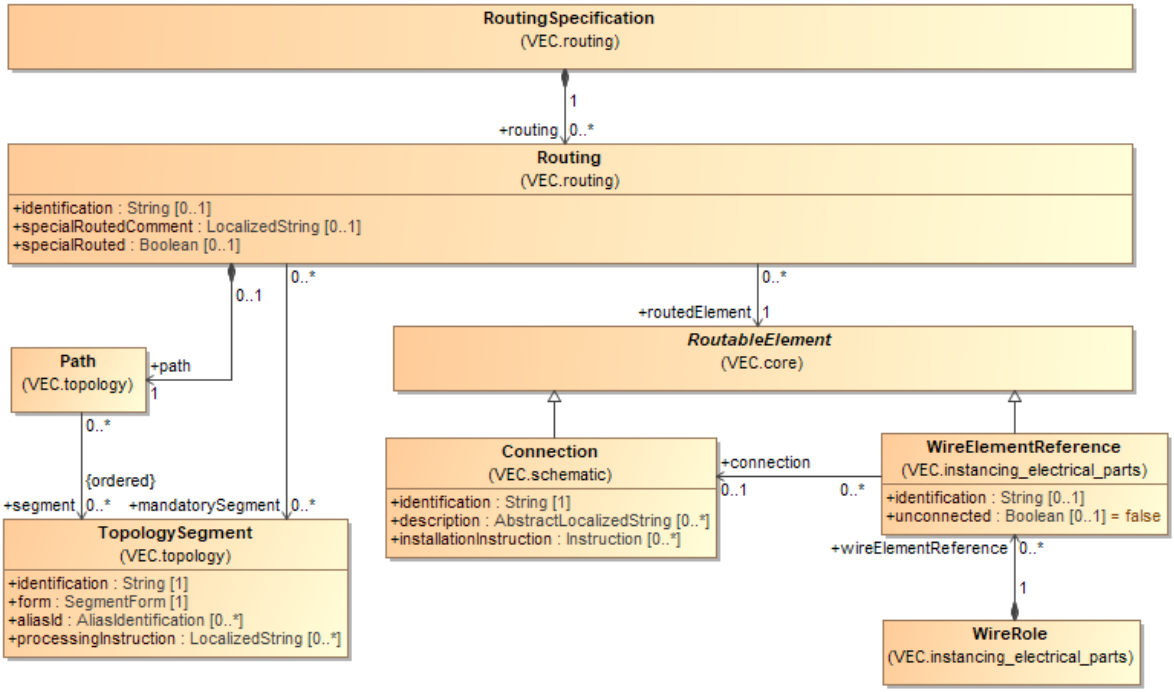


Figure 65: Routing

A *RoutingSpecification* is a container for various *Routings*. Each *Routing* specifies a dedicated *Path* as an ordered list of *TopologySegments* through which the referenced *RoutableElement* is routed. A *RoutableElement* can either be a *Connection* or a *WireElement* (which can alternatively belong to a *PartUsage* or a *PartOccurrence*).

In addition, a *Routing* can reference some segments on the *Path* as *mandatorySegments.* This can be helpful in cases where otherwise a simple redefinition respectively recalculation would not consider these segments.

5.10 Connectivity

5.10.1 Signal Specification

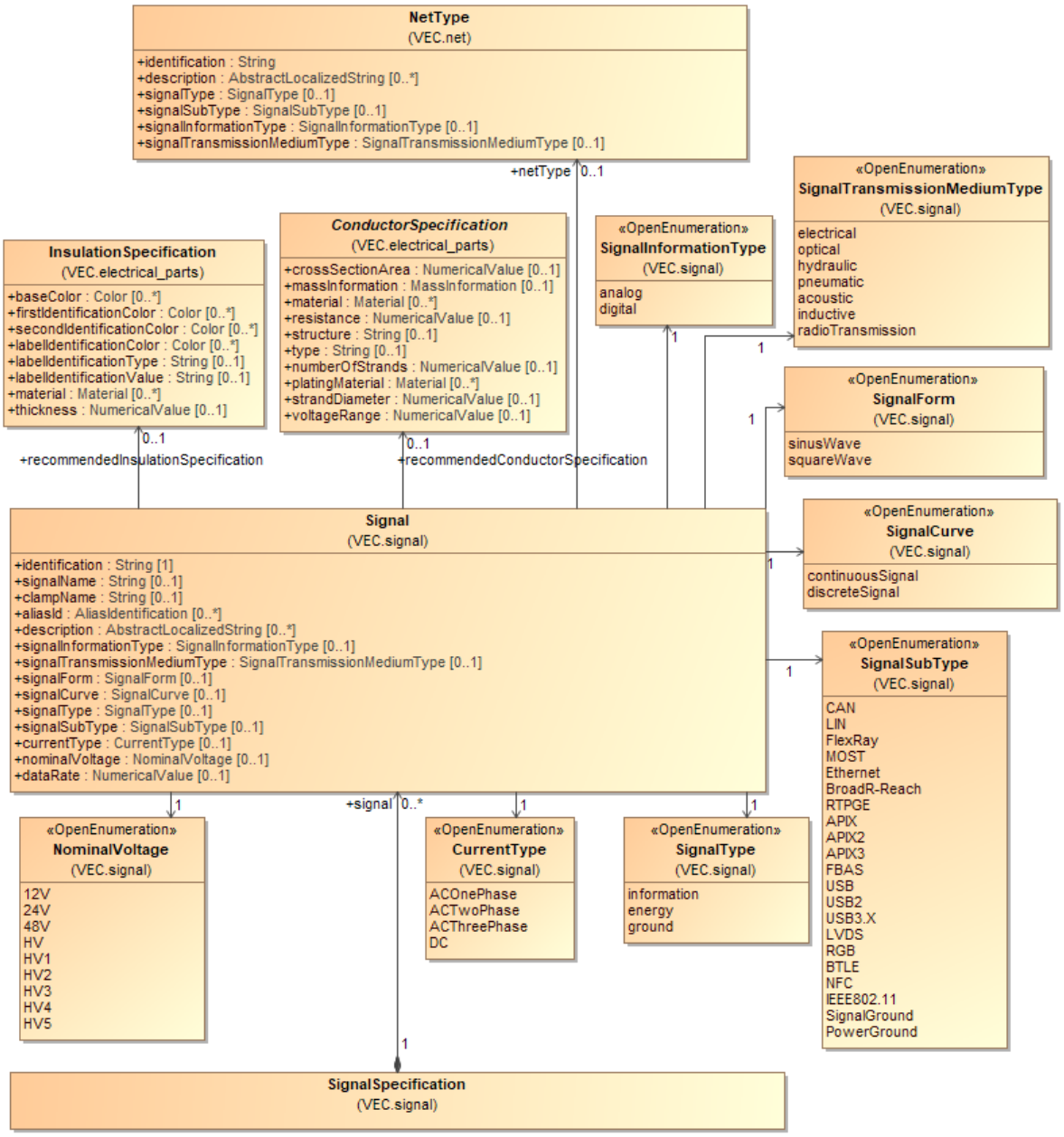


Figure 66: Signal Specification

A *SignalSpecification* defines a set of *Signals* used in the vehicle electrical system. A *Signal* is a more general concept than *Nets*, *Connections* or *Wires,* as they are defined in the scope of a certain vehicle electrical system or a wiring harness. A *Signal* in the VEC represents a physical signal in contrast to abstract messages that are often used to define ECUs and their interfaces (e.g. in AUTOSAR). Typically, the same *Signals* are used in different vehicles or different sections of a wiring harness. A *Signal* can define various properties e.g. the *SignalType* or the *SignalForm*. Furthermore, a *Signal* can define a recommended *ConductorSpecification* and a recommended *InsulationSpecification.* This mechanism can be used for example to provide

information about the recommended color coding of a certain *Signal*, regardless of the vehicle electrical system or wiring harness section where it is implemented.

5.10.2 Net Specification

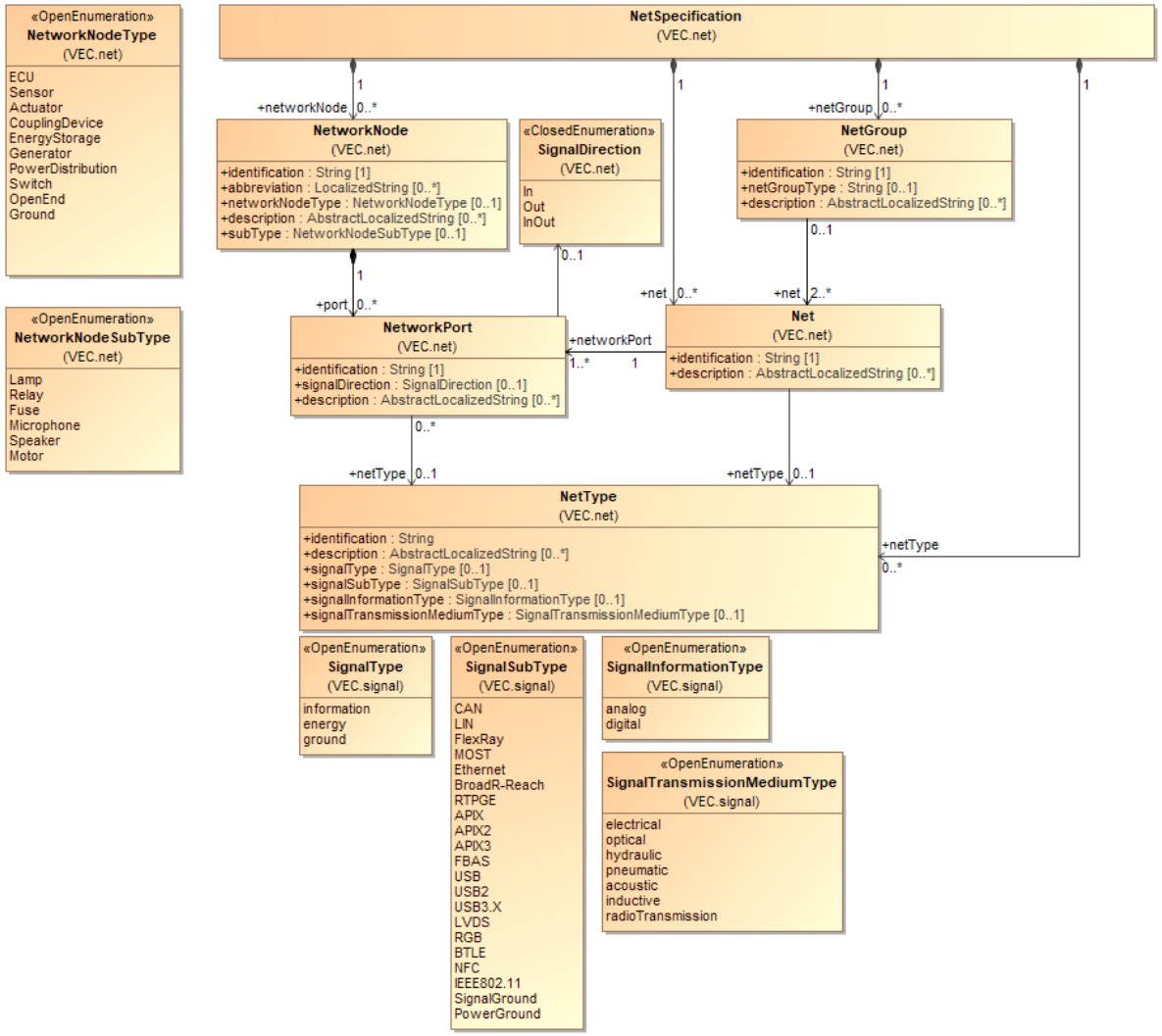


Figure 67: Net Specification

A *NetSpecification* is the most abstract way to describe a vehicle electrical system. It is a container for various *NetworkNodes*, *Nets* and *NetGroups*. A *NetSpecification* is used if the physical links between electrical components are specified without specifying a concrete network topology and a physical realization. In many processes and applications this is defined as "Architectural Layer".

A *NetworkNode* is a representative for an actor in the electric system, e.g. an actuator, a sensor, an ECU. It can define various *NetworkPorts,* which can be classified as a signal source (in the case that the attribute *signalDirection* has got the value *Out*) or as a signal sink (in the case that the attribute *signalDirection* has got the value *In*) or with changing behaviour (in the case that the attribute *signalDirection* has got the value *InOut*).

Note: According to the definition above inliners (normally) and splices are no *NetworkNodes* and so are not represented within a *NetSpecification*. In some cases, there are architectural relevant inliners (NetworkNodeType = CouplingDevice) that represented in the architectural layer.

A *Net* is a representative of an abstract link between the referenced *NetworkPorts* and can be related to a NetType.

Note: A *Net* itself doesn't define:

- How the topology of the conducting connection must be realized (e.g. if three *NetworkPorts* are interconnected, the *Net* makes no definition if this physical three-point connection is realized with a splice, an insulation displacement connector or a double contact.).
- How the physical connections must be realized (e.g. if a certain BUS Connection is realized by a pair of wires or four wires).

Note: Normally, a *Net* references at least two *NetworkPorts*. However, in the very early stages of product development it might be the case that some *NetworkNodes* with *NetworkPorts* which are source of a dedicated Net are already defined – but the counterparts are not. This is an example where it can be useful to define *Nets* referencing only one *NetworkPort*.

5.10.3 Connection Specification

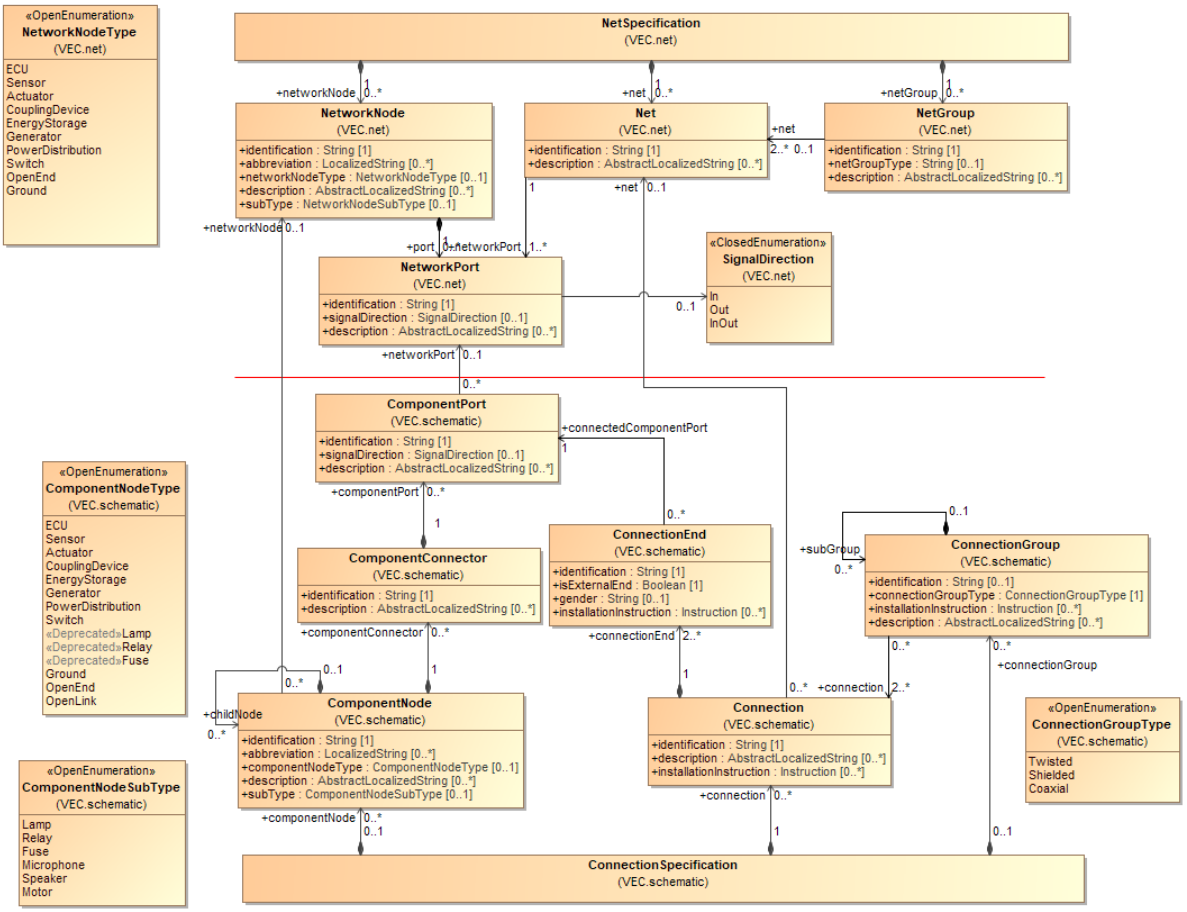


Figure 68: Connection Specification

A *ConnectionSpecification* is intended to contain the logical information that is usually included on an electrical system connection plan (system schematic). It can either stand alone or detail the information of a *NetSpecification*. A *ConnectionSpecification* is a container for various *ComponentNodes*, *Connections* and *ConnectionGroups*. A *ConnectionSpecification* is used, if requirements for realization of the network should be specified (e.g. should a *Net* be realized with a single or multiple connection (BUS-Systems)).

Note: Logical information means a *ConnectionSpecification* doesn't contain layout information. Layout information can be exchanged by other formats like SVG, PDF or DXF.

A *ComponentNode* is a representative for an element in the electric system, e.g. an actuator, a sensor, an ECUs. In this way it is quite similar to a *NetworkNode* and may even reference the corresponding *NetworkNode* in this case. Moreover, a *ComponentNode* can define *childNodes* in order to describe its internal structure.

There is no general rule if inliners shall appear in the system schematic or not. In many cases there is a detail of the physical design of the harness (like splices). Their creation is done implicitly when an electrological design is routed into a specific topology and the routing crosses an inliner definition of the topology. In these cases, the inliner definition should not appear in the system schematic. However, there are other scenarios where the inliner definition is done on purpose and as part of the electrological design (e.g. for providing a pluggable modularity in the system). In these cases an inliner can appear in the *ConnectionSpecification*.

A *ComponentNode* can specify various *ComponentPorts*. Each *ComponentPort* can reference its corresponding *NetworkPort* (if it has one).

Connections represent the physical realization of a *Net* (without a topology). A *Net* (e.g. the BODY-CAN-BUS) can be realized by one or more physical connections (e.g. a CAN-BUS is normally realized by two physical connections (HI & LOW)). The *Connections* do not define a topology of the realization. This means that a CAN *Connection* with three *ConnectionEnds* can be realized in different ways (e.g. three wires with splice, two wires with a double contact or an IDC / IDS contact).

A *ConnectionGroup* references two or more *Connections* expressing the physical realization of the referenced *Connection* shall be somehow grouped e.g. twisted. For complex structures a *ConnectionGroup* can specify subgroups. Finally, a *ConnectionGroup* can reference a *NetGroup* in order to express a refinement-relationship.

5.10.4 Wiring Specification

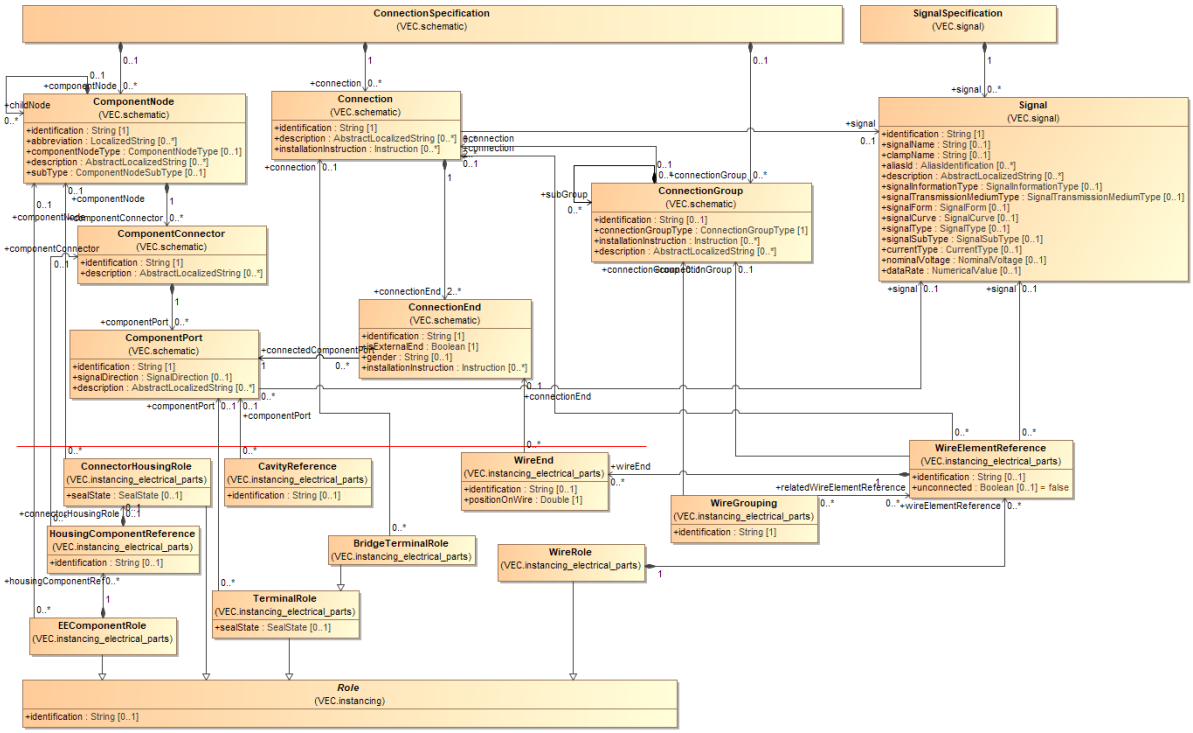


Figure 69: Wiring Specification

The VEC model breaks down the information that is usually included on an electrical system wiring plan into three aspects: part usage information, contacting information and mating information. As part usage information is for instance also relevant in the geometry respectively part placement context and as contacting and mating information is also relevant in the KBL context (the final definition of wiring harnesses and harness modules), the VEC model does not define an integrated WiringSpecification class for all these aspects. Instead, the VEC model defines for each of the three mentioned aspects a separate specification class, which can be reused when necessary (see also "Instances of Components").

EE-components in a wiring diagram are intended to be described as *PartUsages* with an *EEComponentRole*. Each of these *PartUsages* can reference an *EEComponentSpecification* in order to describe relevant technical properties. The *EEComponentRole* can specify a *TerminalRole* for each EE-component pin. This can act as the basis for a mating definition between the EE-component pins and harness connector pins.

Harness connectors in a wiring diagram are intended to be described as *PartUsages* with a *ConnectorHousingRole*. Each of these *PartUsages* can reference a *ConnectorHousingSpecification* in order to describe relevant technical properties.

Harness connector pins in a wiring diagram are intended to be described as *PartUsages* with a *PluggableTerminalRole*. Each of these *PartUsages* can reference a *PluggableTerminalSpecification* in order to describe relevant technical properties. The relationship between harness connector and harness connector pin is intended to

be described by a *ContactingSpecification* which defines an appropriate *CavityMounting*.

Wiring connections are intended to be described as *PartUsages* with a *WireRole* in combination with a *ContactingSpecification* which defines an appropriate *WireMounting*. Each of these *PartUsages* can reference a *WireSpecification* in order to describe relevant technical properties.

The *PartUsages* with *EEComponentRole* respectively *ConnectorHousingRole*, *TerminalRole* or *WireRole* can each reference their specific corresponding element within a *ConnectionSpecification* in order to express a refinement-relationship.

5.10.5 Contacting Specification

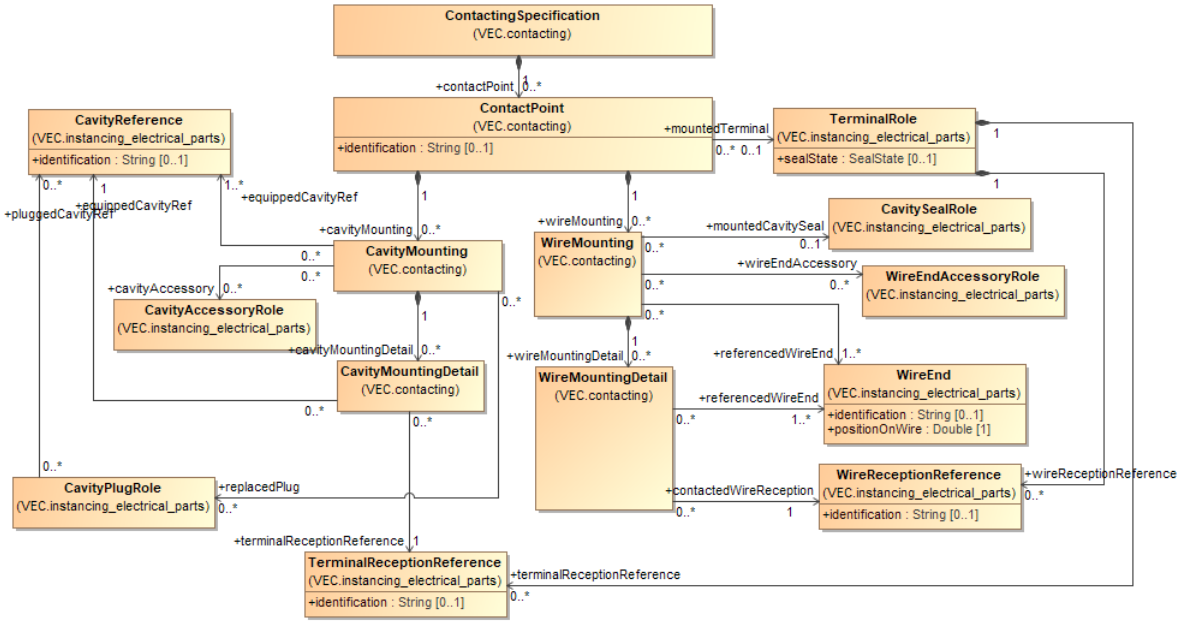


Figure 70: Contacting Specification

A *ContactingSpecification* is used to describe contacting information in several uses cases: It can be used to describe both the more abstract contacting information which is contained on an electrical system wiring plan and the very concrete contacting of an assembly / module/ harness configuration or even a 150% harness description.

A *ContactingSpecification* is a container for various *ContactPoints*. Each *ContactPoint* can specify several *CavityMountings* and *WireMountings* and will in most cases reference a *TerminalRole*.

A *WiringMounting* references one or more *WireEnds* stating they are contacted with the *TerminalRole* referenced by the *ContactPoint*. In cases of ambiguity a *WireMounting* can specify various *WireMountingDetails* in order to describe which *WireEnd* is contacted to which *WireReception* of the referenced *TerminalRole*. Finally, a *WireMounting* can reference a *CavitySealRole*.

A *CavityMounting* references one or more *CavityReferences* expressing they are equipped by the *TerminalRole* referenced by the *ContactPoint*. In cases of ambiguity

a *CavityMounting* can specify various *CavityMountingDetails* in order to describe which *TerminalReception* is mounted in which *CavityReference*. Finally, a *CavityMounting* can reference various *CavityPlugRoles* stating the *CavityMounting* replaces those *CavityPlugs*.

Note: *ContactPoints* are *ConfigurableElements* and therefore can reference a *VariantConfiguration*.

Note: There may be use cases where a *ContactingSpecification* is wanted to describe an incomplete contacting. An example might be an assembly or module with wires together with mounted terminals but without a mounting of the terminals into a connector housing. The missing contacting information (the mounting of the terminals into a connector housings) may be subject of other *ContactingSpecifications* respectively *ContactPoints* which may be specified configuration dependent.

5.10.6 Coupling Specification

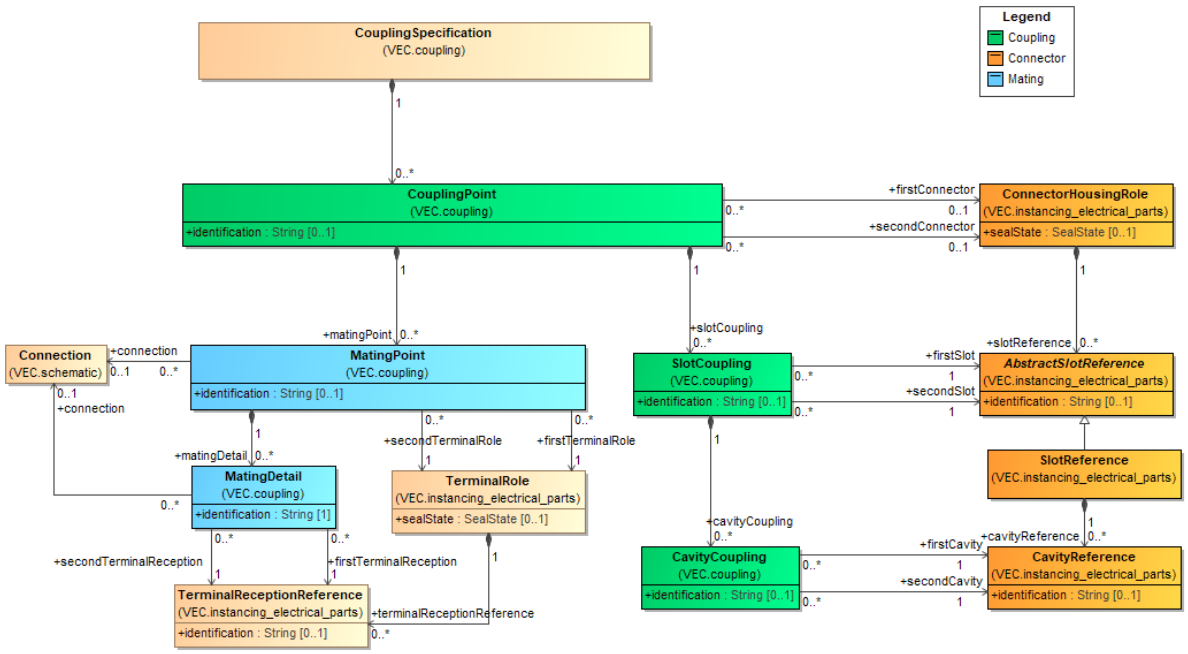


Figure 71: Coupling Specification

A *CouplingSpecification* is a container for various *CouplingPoints*. Each *CouplingPoint* defines a pluggable connection. All sub elements are disconnected if the coupling is disconnected. A coupling can occur with a connector (e.g. Inliner) or without a connector (e.g. ring terminal).

The *CouplingPoint*, the *SlotCoupling* and the *CavityCoupling* defines the mapping between two coupled connectors.

The *MatingPoint* defines the mapping between terminals and therefore the electrical properties of the coupling.

Each *MatingPoint* references two *TerminalRoles* as a pair of terminals that are intended to be connected in the vehicle electric system. This can either be a female

and a male pluggable terminal or a ring terminal and a bolt. In cases of ambiguity a *MatingPoint* can additionally specify various *MatingDetails* in order to reference the relevant *TerminalReceptions*.

Note: *MatingPoints* are *ConfigurableElements* and so can reference a VariantCondition.

In the development process exist multiple scenarios, where a coupling will not be specified completely. E.g. in the electro logical design, a coupling might be defined based on terminals only (e.g. between the wiring harness and an ECU), while the used connector is not yet known. The opposite scenario can occur in the geometric development.

5.10.7 Wire Grouping Specification

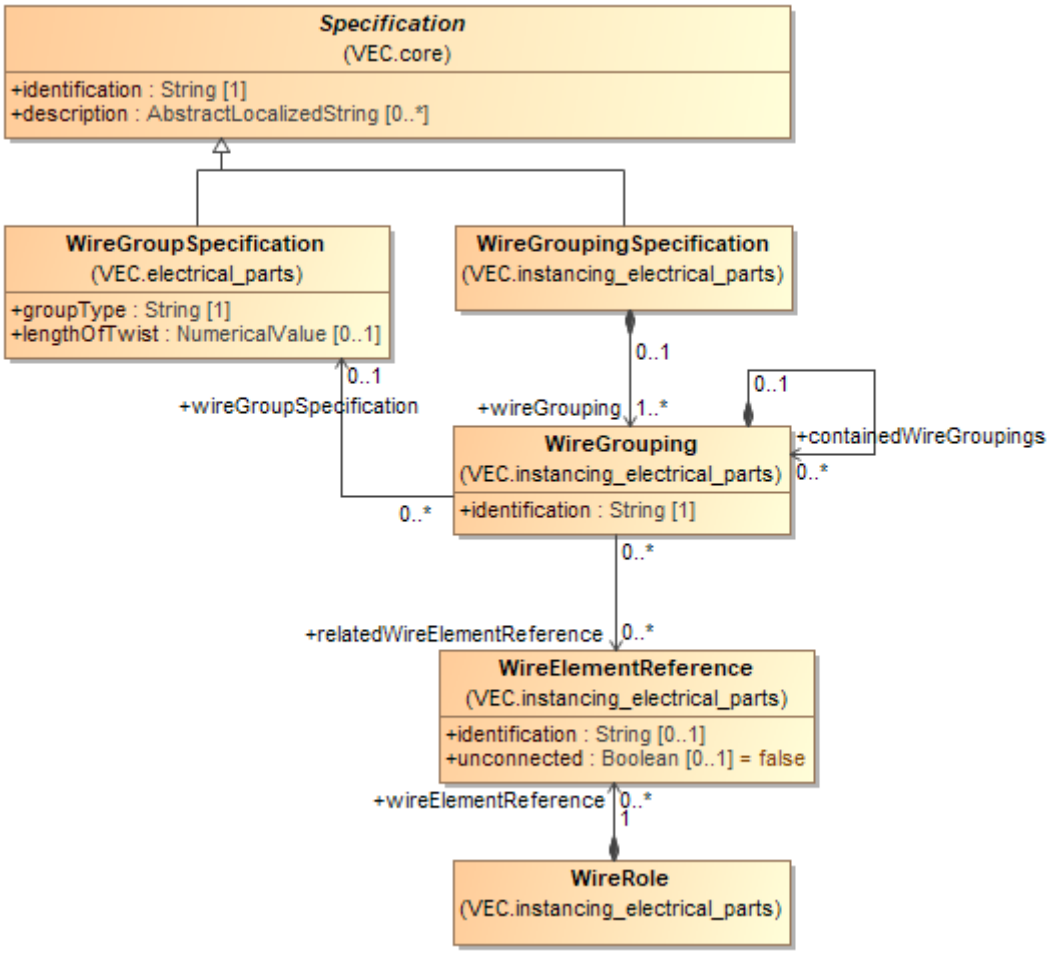


Figure 72: Wire Grouping Specification

A *WireGroupingSpecification* is used to describe grouping requirements (e.g. twisted pair) in a certain context of usage (e.g. in the context of a 150% harness description). For this, a *WireGroupingSpecification* is a container for various *WireGroupings*. For more details, please refer to the description of the *WireGrouping.*

5.10.8 Pin Wire Mapping

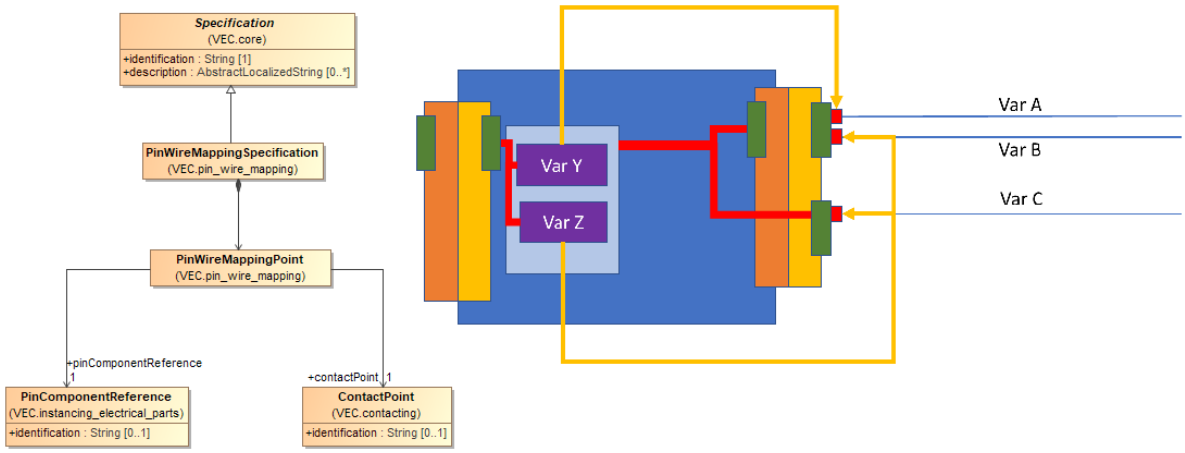


Figure 73: Pin Wire Mapping

A *PinWireMappingSpecification* can be used to create **variance free** mappings between a wire (represented by the contact point) and the pin of an E/E component. This is a possibility to create a shortcut in the model between a wire and its connected E/E-component (e.g. a fuse) that might be only indirectly connected to a wire (e.g. via a fuse and relay carrier). This is a relevant information for e.g. the validation of fusing.

In a concrete 100% variant, this is an information that could be derived from the data provided by the VEC, if available. Informally spoken, the way in the model would be from the *WireEnd* over to the *ContactPoint* and from there to the cavity of the harness connector. From the harness connector to the fuse and relay carrier *PinComponentReference*, via the *CouplingPoint.* From the *PinComponentReference*, via the *InternalComponentConnection* to the slot of the fuse and then via another *CouplingPoint* to the fuse.

However, in a 150% scenario, even if all component assignments to modules are known, this information cannot be derived unambiguously without extensive knowledge of the variant management mechanisms. This is illustrated with the graphic representation in the diagram. There are two wires (controlled by Variant A & B) attached to a single cavity (green). Via an internal connection of the carrier they are connected to a single component slot (light blue). This slot contains two different fuses (controlled by variant Y & Z). Without knowledge of the conditions between the variant A, B, Y & Z it is impossible to say what fuse is responsible for the protection of which wire. That knowledge might not be available at all times, for all partners in the process.

The *PinWireMappingPoint* provides a possibility to define this information in the VEC (the golden / yellow arrows in the illustration).

5.11 External Mapping

The external mapping in the VEC defines a mechanism to provide a link between the content data of the VEC and external data sources (e.g. SVG images, JT models, requirements, ...). The only requirement on the external data source for this mapping

approach is, that the contained data is in some way structured and that the elements are identifiable with a unique key.

The reasons for embedding the mapping information in the VEC are:

- The mapping can be easily read and recreated together with VEC and references to VEC elements can be followed in a simple way since the mapping can reference the elements directly within a single file (IDREF).
- No additional format definition is necessary.

5.11.1 External Mapping

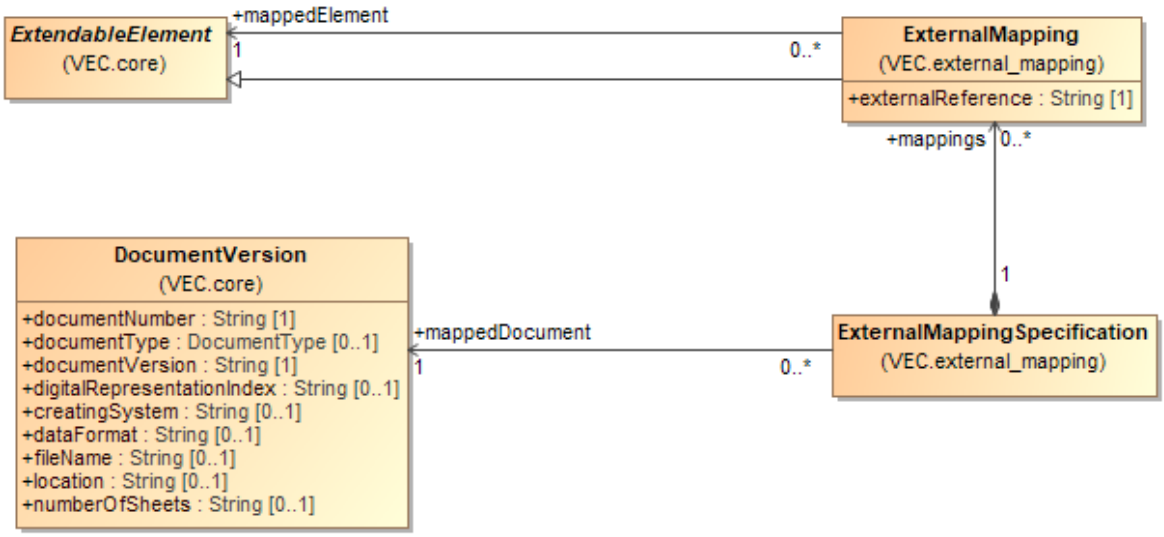


Figure 74: External Mapping

The diagram displays the extensions in the VEC Modell necessary to support a mapping to external sources. The extension consists of two classes *ExternalMappingSpecification* and *ExternalMapping*. An *ExternalMappingSpecification* defines a mapping for a single *DocumentVersion* specified by the association *mappedDocument*. The referenced *DocumentVersion* itself identifies by its attributes the external document which should be mapped (e.g. an SVG-File, a requirements-document).

The mapping of the individual elements in the VEC is defined with the class *ExternalMapping*. It points to an *ExtendableElement* (which can be almost any VEC-Element) as *mappedElement*. The attribute *externalReference* contains the unique key of the same element in the context of the mapped *DocumentVersion*.

The reason for defining the *ExternalMapping* elements within a separate *Specification* is that the *ExternalMappingSpecification* can be placed in the VEC within its own *DocumentVersion.* The creation of mapping does not require a modification and thus a versioning of the mapped content in the VEC.

5.12 XML Representation of the Model

5.12.1 VEC-Root

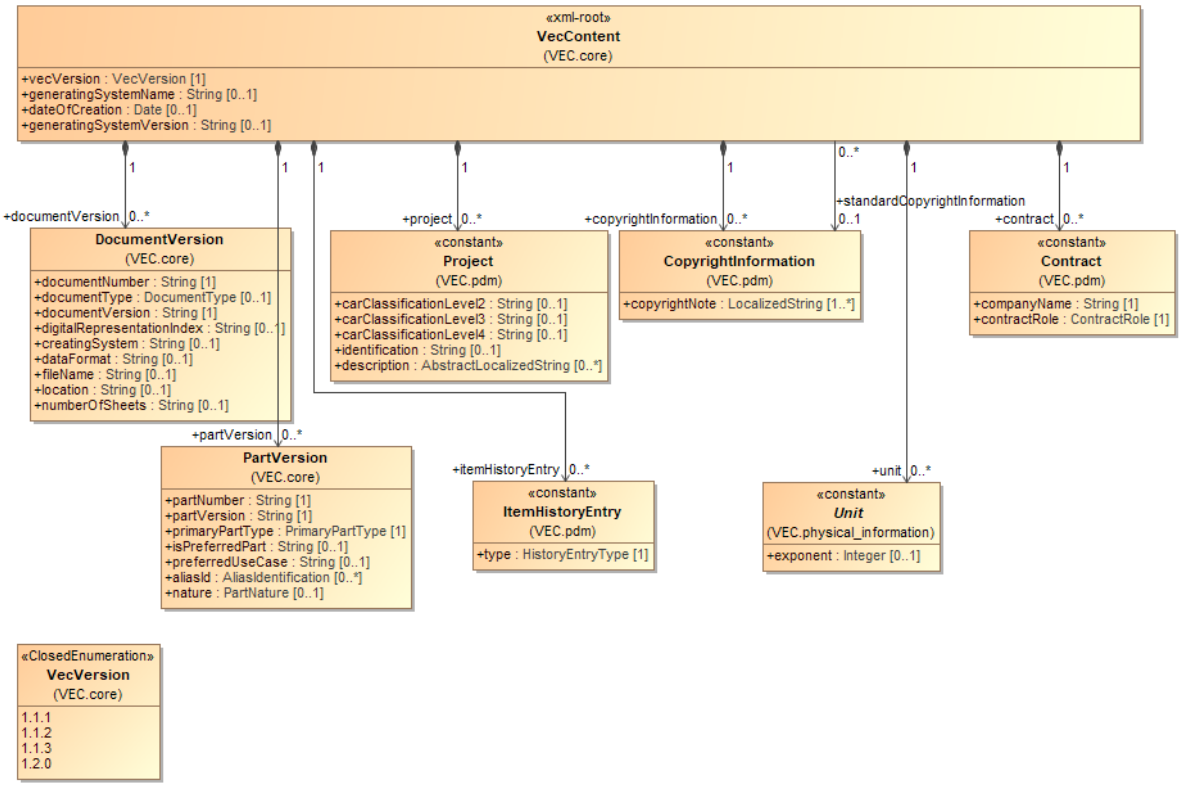


Figure 75: VEC-Root

The diagram above defines the class *VecContent* which is intended to be regarded as the top-level root element for VEC documents. It is the declared entry point to the information that a VEC file contains.

A *VecContent* is a container for various *DocumentVersions*, *PartVersions*, *Projects*, *ItemHistoryEntries*, *CopyrightInformations*, *Units*, and *Contracts. DocumentVersion* and *PartVersion* are the only two classes in the VEC model that define and contain version information. Almost all information a VEC document can contain is subordinated to one of those two classes. This means, that every piece of information which is represented as an instance of a class of a certain attribute value is distinctly related to a dedicated *PartVersion* or *DocumentVersion*.

The only exceptions are classes with the stereotype <<*constant*>> as shown in the diagram.

5.12.2 Mapping of the VEC Model to XML schema definition (XSD)

The mapping of the VEC model to an XML schema definition (XSD) described in the following defines a standardized syntax for the exchange of harness design data across process steps and supporting tools.

The dedicated namespace prefix of the schema definition is “vec”, the namespace is "http://www.prostep.org/ecad-if/2011/vec".

Each VEC model class that does not contain the stereotype <<*enumeration*>> is mapped to an XSD complex type with the same name as the VEC model class name. If the VEC model class is a derived subclass the corresponding XSD complex type uses the XSD extension base construct. Not inherited VEC model class attributes are mapped to a sequence of XSD elements with the same names as the names of the corresponding VEC model class attributes. The cardinality and data type definitions are defined equivalent to the VEC model. The base data types are:

- *xs:string* (corresponding to the VEC model data type definition *String*)
- *xs:dateTime* (corresponding to the VEC model data type definition *Date*)
- *xs:integer* (corresponding to the VEC model data type definition *Integer*)
- *xs:double* (corresponding to the VEC model data type definition *Double*)
- *xs:boolean* (corresponding to the VEC model data type definition *Boolean*)

Compositions in the VEC model are handled in the same way as if the aggregated class would have been modelled as a class attribute of the owning class. The name definition corresponds to the target role name.

As all associations in the VEC model are directed they are mapped to an attribute which is owned by the XSD complex type that relates to the VEC model class that is source of the association. The name is equal to the name of the association target role name. The type is xs:IDREF respectively xs:IDREFS depending on the association target cardinality.

As associations are mapped to xs:IDREF respectively xs:IDREFS, each XSD complex type that is not a refinement of another XSD complex type (by usage of the XSD extension base construct) defines an additional required attribute with the name “id” and the type xs:ID. This attribute is introduced as reference point for links. The values for the “id”-attributes are normally not defined by a user. Exporters are expected to generate valid values for the “id”-attributes so that in the end

- Links (as instances of associations) are expressed correctly
- The resulting XML file is XSD compliant.

Note: The values for the “id”-attributes have temporary character. By this recommendation exporters are not required to remember values they have once used during an export. Exporters are explicitly allowed to assign different values for the “id”-attributes during every export process.

The XSD defines a VecContent element as xs:element. This is the declared root element for VEC compliant XML-documents. A VEC compliant XML-document is limited to one instance of a VecContent.

5.12.3 Partitioning and Sizing of VEC Files

In theory it is possible to export all data of a company that relates to the vehicle network in a single VEC file. However, this is not a very reasonable approach for several reasons. Amongst those are:

1. *Intellectual Property Protection:* For everything that is contained in a single VEC file, the "All or Nothing Principle" applies and it is hard or impossible to enforce a "Need to Know". A recipient of a file, who requires only a single detail, always receives the complete information.
2. *Partial Changes:* Even if only a single content element is changed, the complete file must be created and read, because only after reading the complete file the recipient knows which elements have changed.
3. *Tool Support:* Tools must support large areas of the VEC in their Import- / Export interfaces, even if the areas do not belong to their use case.
4. *Technical Feasibility:* If all information is contained in a single VEC file, the file itself can become huge, which places new requirements on the creating and reading tools (e.g. required memory).

Therefore, the general recommendation is to make a VEC files **as small as possible and as large as necessary**. The correct scoping of a generated VEC file depends on the use case of the information and the intended interface. Scoping means in this context the process of defining which information shall be bundled together in a single file for data exchange.

The following paragraphs contain some guidelines if information shall be packaged together or not.

1. If the elements specified with the VEC are described, published and changed independently from each other regarding time and content, then they shall be placed in separate files (unit of publication). For example, this is the case for the description of harness components (connectors, wires etc.), represented by a part number. This rule applies only for the publication as master information from the original source of information. If the information is used to create other information, it can be embedded in a single file. For example, the description (complete or partial) of the used harness components will be embedded in the VEC describing a harness.
2. If elements specified with the VEC have no relationships between each other (except reference base on *PartVersions* and *DocumentVersions*), then they shall be placed in separate files. For example, the specifications of a connector housing and a wire have no reference.
3. If the relationship between two elements is indirectly over shared information, then this is **no** reason for the elements themselves to be placed in a single file (e.g. two connectors share the same *CavitySpecification* or two wires share the same *CoreSpecification*). If this piece of shared information is defined centrally, then it would have its own unit of publication, probably in its own *DocumentVersion*. In this case rule #1 can be applied. This means that, if the specification (e.g. a *CoreSpecification*) shall be exchanged, it shall be placed in its own VEC file. If the specification is used to describe another element (e.g. a wire) it shall be embedded in the VEC file of the described element. However, this is no reason to place all elements using this information in the same VEC file.

6 Appendix A: Glossary

ECAD	electronic computer aided design
ECU	electronic control unit
GEO	A data format specification for the description of geometrical data in the wiring harness context.
IDC	insulation displacement connector
Item	In the context of this recommendation an item is either a part or a document.
KBL	harness description list (“Kabelbaumliste”). A data format specification for the description of wiring harness data.
prostep ivip	an international association that has committed itself to developing innovative approaches to solving problems and modern standards for product data management and virtual product creation.
XML	Extensible Markup Language
XSD	XML schema definition
VDA	German Association of the Automotive Industry (“Verband der Automobilindustrie”)
VEC	Vehicle Electric Container. A data format specification for the description of harness design data cross process steps and supporting tools.

7 Appendix B: Data Model Description

7.1 Module assignment_groups

7.1.1 Class AssignmentGroup

An *AssignmentGroup* is a concept that allows the clustering of arbitrary elements in ways that are orthogonal to hierarchical and semantic structure of the VEC.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the AssignmentGroup. The identification is guaranteed to be unique within the specification.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the *AssignmentGroup*.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConfigurableElement	0..*	associatedAssignmentGroups	0..*	N	
AssignmentGroupSpecification	1	assignmentGroup	0..*	Y	Contains the AssignmentGroups that are defined by this AssignmentGroupSpecification.

7.1.2 Class AssignmentGroupSpecification

Specification that contains *AssignmentGroups*.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
AssignmentGroup	assignmentGroup	0..*	1	Y	Contains the AssignmentGroups that are defined by this AssignmentGroupSpecification.

7.1.3 Class DocumentRelatedAssignmentGroup

A DocumentRelatedAssignmentGroup allows the creation of traceability links to elements in a *DocumentVersion* for a set of VEC objects. The semantic of the traceability link is defined by the *DocumentRelationType.*

General Information

Base Classifier	AssignmentGroup
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	DocumentRelationType	0..1	
relatedIdentification	String	0..1	If this group relates to a specific element in the *relatedDocumentVersion* the identifier of the element is defined in this attribute (e.g. a requirements number).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
SheetOrChapter	relatedSheetOrChapter	0..1	0..*	N	Allows a more specific relationship to a *SheetOrChapter* within the *relatedDocumentVersion.*
DocumentVersion	relatedDocumentVersion	1	0..*	N	References the *DocumentVersion* to which this group relates.

7.1.4 Class FunctionalAssignmentGroup

The *FunctionalAssignmentGroup* clusters elements that contribute to a specific function or a functional aspect. With such a group, certain functional requirements can be associated.

General Information

Base Classifier	AssignmentGroup
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
functionalRequirements	FunctionalRequirement	0..*	Functional requirements that apply to the members of this group.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
FunctionalStructureNode	0..*	containedGroups	0..*	N	

7.1.5 Class FunctionalRequirement

Allows the definition of functional requirements in an enumerable way (e.g. conformance to a certain ASIL level). Attributes of this type have the multiplicity of [0..*]. The following restrictions apply:

- For a combination of type & referenceSystem only a single value is allowed. For single type
- For a specific type and different references systems, multiple values are allowed. However, they must express the same semantic value.
- For different types multiple values are allowed.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	FunctionalRequirementType	0..1	The type defines to which category a requirement belongs (e.g. Functional Safety).
referenceSystem	String	1	The reference system identifies the system in which the values are defined (e.g. ISO26262)
value	String	1	The value that represents the functional requirement (e.g. ASIL D).

7.1.6 Class FunctionalStructureNode

FunctionalStructureNodes can be used to define a hierarchical structure on *FunctionalAssignmentGroups*. Every *FunctionalStructureNode* can reference *FunctionalAssignmentGroups* and *FunctionalStructureNodes* as children.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the *FunctionalStructureNode.*
aliasId	AliasIdentification	0..*	Room to specify additional identifiers for the *FunctionalStructureNode*.
description	AbstractLocalizedString	0..*	On optional human readable description of the *FunctionalStructureNode*.
abbreviation	LocalizedString	0..1	Room for a human readable short name, title etc. of the *FunctionalStructureNode*.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

FunctionalAssignmentGroup	containedGroups	0..*	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
FunctionalStructureSpecification	0..1	rootNode	1	Y	
FunctionalStructureNode	0..1	childNodes	0..*	Y	

7.1.7 Class FunctionalStructureSpecification

Specification to define any hierarchical structure on functional assignment groups (e.g. by the means of a functional organization). The hierarchy starts with a single root node.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
FunctionalStructureNode	rootNode	1	0..1	Y	

7.1.8 Enumeration DocumentRelationType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
ResponsibilityLink		
RequirementsLink		

7.1.9 Enumeration FunctionalRequirementType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
FunctionalSafety		Functional safety requirements e.g. ASIL Level.

Crash		Crash requirements e.g. post-crash functionality.
Legal		Legal Requirements.
Function		Requirements from a functional point of view (e.g. standby current).

7.2 Module contacting

7.2.1 Class CavityMounting

A CavityMounting defines the cavities (CavityReference) where the contacted elements (Terminal) will be mounted.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavityAccessoryRole	cavityAccessory	0..*	0..*	N	
CavityReference	equippedCavityRef	1..*	0..*	N	References the cavities that are used for the cavity mounting.
CavityPlugRole	replacedPlug	0..*	0..*	N	References the cavity plugs that are obsolete if the cavity mounting is realized.
CavityMountingDetail	cavityMountingDetail	0..*	1	Y	Specifies the CavityMoutingDetails, if a detailed description of the relationships between Cavities and TerminalReceptions is needed.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ContactPoint	1	cavityMounting	0..*	Y	Defines the mounting to a cavity of the terminal associated with the ContactPoint. The cardinality is 0..* in order to allow a variant dependant cavity mounting. In such a scenario a cavity mounting is valid in a configuration if all addressed cavities and therefore the associated connector housing is available.

7.2.2 Class CavityMountingDetail

With a CavityMountingDetail it is possible to describe a detailed cavity mounting.

This is needed if the information which terminal reception is mounted into which cavity is important. There are cases where this information can be relevant (e.g. bridge contacts with an asymmetric wire mounting).

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavityReference	equippedCavityRef	1	0..*	N	References the cavity that is used for the detailed description of the cavity mounting.
TerminalReception Reference	terminalReception Reference	1	0..*	N	References the TerminalReception that is used for the detailed description of the cavity mounting.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CavityMounting	1	cavityMountingDetail	0..*	Y	Specifies the CavityMoutingDetails, if a detailed description of the relationships between Cavities and TerminalReceptions is needed.

7.2.3 Class ContactingSpecification

Specification for the description of the contacting. A contacting defines the relationships between Terminals, Seals, Plugs, Cavities and Wires.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ContactPoint	contactPoint	0..*	1	Y	Specifies the ContactPoints defined by the ContactingSpecification.

7.2.4 Class ContactPoint

A contact point defines the relationship between Terminals, Seals, Plugs, Cavities and Wires. It specifies a single contacting variant. This means that the contacting is manufactured, as specified by the *Contact*Point. Either all participants (Cavities, Terminals, Seals, Wires) set into a relationship by the *ContactPoint* exist in a specific harness or none. There is no requirement, to filter the participants of a contacting situation with information derived from VariantConfigurations or assembly / module associations in order to create a manufacturing variant.

The *ContactPoint* represents a single potential. Consequently, all cavities and wires referencing / being referenced by a *ContactPoint* are short-circuited and have the same potential (even if the signals on the wires are named differently. If a contacting of a terminal has more than one potential (e.g. a coax-contact) one contact point for each potential is needed.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the ContactPoint. The identification is guaranteed to be unique within the ContactingSpecification.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireMounting	wireMounting	0..*	1	Y	Specifies the WireMouting defined by ContactPoint. More than one WireMounting is allowed in order to support variance. In concrete configuration the WireMounting with all referenced elements present is used.
CavityMounting	cavityMounting	0..*	1	Y	Defines the mounting to a cavity of the terminal associated with the ContactPoint. The cardinality is 0..* in order to allow a variant dependant cavity mounting. In such a scenario a cavity mounting is valid in a configuration if all addressed cavities and therefore the associated connector housing is available.
TerminalRole	mountedTerminal	0..1	0..*	N	References the terminal that is used for contacting defined by the ContactPoint.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ContactingSpecification	1	contactPoint	0..*	Y	Specifies the ContactPoints defined by the ContactingSpecification.
PinWireMappingPoint		contactPoint	1	N	

7.2.5 Class WireMounting

A *WireMounting* defines the mounting of a wire end to terminal. If the contacting is required to be waterproof a cavity seal can be mounted additionally.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End	This	General

Type	Role	Mult	Mult	Agg	Comment
WireEndAccessoryRole	wireEndAccessory	0..*	0..*	N	
WireEnd	referencedWireEnd	1..*	0..*	N	References the wire ends that are used for the wire mounting. The minimum cardinality is one, because a wire mounting without wire end makes no sense. The maximum cardinality is * in order to support multi crimps.
WireMountingDetail	wireMountingDetail	0..*	1	Y	Specifies the WireMoutingDetails, if a detailed description of the relationships between WireEnds and WireReceptions is needed.
CavitySealRole	mountedCavitySeal	0..1	0..*	N	References the cavity seal that is used for the wire mounting.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ContactPoint	1	wireMounting	0..*	Y	Specifies the WireMouting defined by ContactPoint. More than one WireMounting is allowed in order to support variance. In concrete configuration the WireMounting with all referenced elements present is used.

7.2.6 Class WireMountingDetail

With a WireMountingDetail it is possible to describe a detailed wire mounting.

This is needed if the information which wire end is mounted onto which wire reception is important (e.g. coax contacts).

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireReceptionReference	contactedWireReception	1	0..*	N	References the WireReception that is used for the WireMounting.
WireEnd	referencedWireEnd	1..*	0..*	N	References the WireEnds that are mounted to referenced WireReception. A cardinality of more than one is allowed in order support parallel connectors, where multiple wire ends are placed on one side of the connector (one wire reception) and the other wire ends are placed on the other side of the connector (the other wire reception).

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

WireMounting	1	wireMountingDetail	0..*	Y	Specifies the WireMoutingDetails, if a detailed description of the relationships between WireEnds and WireReceptions is needed.

7.3 Module core

7.3.1 Class AbstractLocalizedString

Abstract super-class for Localized text values.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
languageCode	LanguageCode	1	References the corresponding languageCode of the value.
value	String	1	The value of the LocalizedString in language defined by the languageCode.

7.3.2 Class AliasIdentification

Class for the definition of alias identifications. Alias identifications are additional identifications for VEC-elements, which are valid in a certain scope (e.g. PPS-ID's).

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identificationValue	String	1	Specifies the identification value.
type	AliasIdentificationType	0..1	Defines the type (or the role) of the AliasIdentification. Defined literals are contained in an OpenEnumeration.
scope	String	0..1	The scope in which the AliasIdentification is valid / or the issuer of the alias id. This could be for example a certain process, a company or an IT-System.
description	AbstractLocalizedString	0..*	On optional human readable description of the AliasIdentification.

7.3.3 Class BaselineSpecification

A *BaselineSpecification* defines a set of *ItemVersion*s (*Document-* and *PartVersions*) that relate to each other in a certain way e.g. all parts and documents in their specific versions that contributed to a specific manufactured result.

Baselines are a standard mechanism to support change, release and configuration management.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
state	BaselineState	0..1	Defines the state of the baseline itself (e.g. if it is finalized or work in progress).
content	BaselineContent	0..1	Defines the state of the content of the baseline in regard of its defined scope.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ItemVersion	validVersions	0..*	0..*	N	References the *ItemVersions* that are the content of the baseline.

7.3.4 Class ConfigurableElement

Abstract base class for all elements which can be configured with a VariantConfiguration.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
AssignmentGroup	associatedAssignmentGroups	0..*	0..*	N	
ApplicationConstraint	applicationConstraint	0..*		N	References the application constraints that apply to the ConfigurableElement.
VariantConfiguration	configInfo	0..1	0..*	N	References the configuration information that applies to the ConfigurableElement.

7.3.5 Class DocumentVersion

The DocumentVersion is one of the two anchors for PDM information in the VEC. All technical information about a PartVersion is contained in one or more documents. The documents are containing the actual content of the VEC since all Specifications are an element of a document.

General Information

Base Classifier	ItemVersion
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
documentNumber	String	1	The documentNumber is the major identifier of a DocumentVersion. The format is user defined and respectively company specific.
documentType	DocumentType	0..1	The type of the document, that is defined in an *OpenEnumeration* and gives a hint about the content of the document. Values for typical types of documents in the process are predefined (e.g. a part master document for the specification of a *PartVersion*). At later point, further constraint might be attached to *documentType* defining a minimum content for certain types of documents.
documentVersion	String	1	The documentVersion specifies the version index of a document (see also documentNumber).
digitalRepresentationIndex	String	0..1	An arbitrary change index that indicates if the digital representation (the content in VEC) of this *DocumentVersion* has been changed / regenerated. This can be for example an index, a timestamp or a checksum. This allows the detection of changes in the content, even when the DocumentNumber & DocumentVersion is the same. For a more detailed explanation in the context see "Parts & Documents". KBLFRM-837.
creatingSystem	String	0..1	The creatingSystem specifies the computer application or the machine which is used to create the document.
dataFormat	String	0..1	The dataFormat specifies the convention that was used to structure the information in the document. This is useful if the DocumentVersion is a pointer to an external document, which is not contained in the VEC or if the content of this DocumentVersion was automatically generated by the extraction of the information out of the original document.
fileName	String	0..1	The name of the file as it appears in the VEC-Package, including the folder structure (fully qualified name) that contains this *DocumentVersion*. If this DocumentVersion is a link to an external document (e.g. a ComponentSymbol), then the fileName attribute points to the file containing the original document. The usage of this attribute is only valid, if the original document is distributed along with the VEC-file in a VEC-Package. It must not point to any file location which is not part of the VEC-Package (e.g. a File on a central server file share).
location	String	0..1	The location is a possibility to provide a reference to the source location of the *DocumentVersion* (e.g. a document management system or an archive system) where the original document can be found. The location shall be provided either as an URN or URL.
numberOfSheets	String	0..1	The number of sheets contained in the document.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Specification	specification	0..*	0..1	Y	Specifies the Specifications contained in the DocumentVersion. All structured, technical information in the VEC is described by such Specifications.
SheetOrChapter	sheetOrChapter	0..*	1	Y	Specifies SheetOrChapters defined in this DocumentVersion. These are especially useful if the DocumentVersion represents an external reference.
PartVersion	referencedPart	0..*	0..*	N	The association is an informative link which PartVersions are described by the DocumentVersion.
ItemEquivalence	itemEquivalence	0..*	1	Y	Specifies ItemEquivalances defined by the DocumentVersion.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ExternalMappingSpecification	0..*	mappedDocument	1	N	Reference to the *DocumentVersion* that represents the external data source that connected to the VEC content by the *ExternalMappingSpecification*.
DocumentVersion	0..*	relatedDocument	0..*	N	The association is an informative link which DocumentVersion are related to each other (e.g. by derivation, A Harness-Drawing is related to a 3D-Model).
DocumentBasedInstruction	0..*	referencedDocument	1	N	References the DocumentVersion that is used as an Instruction.
DocumentRelatedAssignmentGroup	0..*	relatedDocumentVersion	1	N	References the *DocumentVersion* to which this group relates.
ExtendableElement	0..*	referencedExternalDocuments	0..*	N	This association allows all *ExtendableElements* in the VEC to reference *DocumentVersions* as "external reference". This association shall be used for the extension of elements in the VEC with information that cannot be represented in the VEC in an appropriate way but can be expressed in some external format (e.g. a specific symbol for a *ComponentNode*). D*ocumentVersions* referenced by this association shall not contain any *Specifications.* This association is no replacement for associations with a more precise semantic like the *DocumentBasedInstruction* or the associations between *PartVersion* and *DocumentVersion.*
VecContent	1	documentVersion	0..*	Y	Specifies the DocumentVersions contained in the VEC-file.
RequirementsConformanceStatement			1	N	References the *DocumentVersion* that contains the requirements to which a conformance statement shall be expressed.

7.3.6 Class ExtendableElement

Abstract base class for extendable elements. Extendable elements have the possibility to define non-standard custom properties.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	true

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
DocumentVersion	referencedExternal Documents	0..*	0..*	N	This association allows all *ExtendableElements* in the VEC to reference *DocumentVersions* as "external reference". This association shall be used for the extension of elements in the VEC with information that cannot be represented in the VEC in an appropriate way but can be expressed in some external format (e.g. a specific symbol for a *ComponentNode*). D*ocumentVersions* referenced by this association shall not contain any *Specifications.* This association is no replacement for associations with a more precise semantic like the *DocumentBasedInstruction* or the associations between *PartVersion* and *DocumentVersion.*
CustomProperty	customProperty	0..*	1	Y	Specifies the CustomProperties of the ExtendableElement.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ExternalMapping	0..*	mappedElement	1	N	

7.3.7 Class ItemVersion

Abstract super-class for physical objects (e.g. a Terminal), virtual objects (e.g. a 150% Harness) as well as documents (e.g. a wiring diagram). In difference to AP 212 the VEC makes it only possible to describe/exchange information about Versions since Master-Objects cannot exist without one or more Versions.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment

abbreviation	LocalizedString	0..*	Room for a short name of the Item. In case of a document the attribute is wanted to contain its title.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the ItemVersion. e.g. Buchsengehäuse 26-polig
companyName	String	1	Defines the publishing company of the ItemVersion. The companyName is part of the main identifier of an ItemVersion together with the corresponding number (partNumber or documentNumber) and version (partVersion or documentVersion).
processingInstruction	Instruction	0..*	Processing instructions for the application of the part or the document.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Approval	approval	0..*	1	Y	Specifies the approval information of the ItemVersion.
Creation	creation	0..1	1	Y	Specifies the information about the creation of the ItemVersion.
ChangeDescription	changeDescription	0..*	0..1	Y	Specifies the change history of the ItemVersion.
CopyrightInformation	copyrightInformation	0..1	0..*	N	References the *CopyrightInformation* that is in effect for this *ItemVersion.* If no *CopyrightInformation* is referenced by the *ItemVersion*, the *CopyrightInformation* that is referenced by the *VecContent* (if defined) shall be considered as in effect for this *ItemVersion.*
Contract	contract	0..*	0..*	N	References the contracts that apply to an ItemVersion.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ItemEquivalence	0..*	item	2..*	N	References all ItemVersion that are considered to be equivalent by the ItemEquivalence. A single *ItemEquivalence* shall only reference *ItemVersions* of the same class (either *DocumentVersions* or *PartVersions*).
BaselineSpecification	0..*	validVersions	0..*	N	References the *ItemVersions* that are the content of the baseline.
ItemHistoryEntry	0..*	successorVersion	1	N	References the ItemVersion that is the successor in the ItemHistoryEntry.
ItemHistoryEntry	0..*	predecessorVersion	1	N	References the ItemVersion that is the predecessor in the ItemHistoryEntry.

7.3.8 Class LocalizedString

Allows the internationalization of text contents. Attributes of the type LocalizedString normally have the multiplicity [0..*]. This means that such an attribute can have multiple values for different locales. It must not have multiple values for the same locale.

General Information

Base Classifier	AbstractLocalizedString
Applied Stereotype	
Is Abstract	false

7.3.9 Class LocalizedTypedString

Allows the internationalization of text contents in a typed way. Attributes of the type LocalizedTypedString normally have the multiplicity [0..*]. This means that such an attribute can have multiple values for different locales and types. It must not have multiple values for the same locale and type.

General Information

Base Classifier	AbstractLocalizedString
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	LocalizedTypedStringType	1	Defines the *type* of the *LocalizedTypedString*. This allows the definition of a more detailed semantic than the semantic of the attribute itself with the type *LocalizedTypedString.* Agreed type values are defined in an OpenEnumeration.

7.3.10 Class PartOrUsageRelatedSpecification

Base class for all specifications which are describing a *PartVersion* or a *PartUsage*. A *PartOrUsageRelatedSpecification* specifies a certain aspect of the described part or usage (e.g. general technical part information, connector housing aspects or wire aspects).

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
specialPartType	String	0..1	The specialPartType allows the specification of subclassifications for a PartOrUsageRelatedSpecification (e.g. different types of connector housings).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartVersion	describedPart	0..*	0..*	N	References the PartVersion(s) to which the information defined in this specification applies. Example: If the PartOrUsageRelatedSpecification is a GeneralTechnicalPartSpecifcation and it defines that the color is "green" then all PartVersion referenced by this association are "green".

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PartUsage	0..*	partOrUsageRelatedSpecification	0..*	N	References the PartOrUsageRelatedSpecification(s) that describe the PartOrUsageRelatedSpecification. KBLFRM-399
SpecificRole	0..*	specification	1	N	References the *PartOrUsageRelatedSpecification* that is instantiated by this *SpecificRole.*

7.3.11 Class PartSubstitutionSpecification

A *PartSubstitutionSpecification* defines a set of *PartVersions* that can be used alternatively, due to an incomplete specification for the 150% product description. For a concrete wiring harness only on valid *PartVersion* remains. The selection logic for valid *PartVersions* is not included in the VEC.

A *PartSubstitutionSpecification* can be used for example tubes or ring terminals, where a part of the specification is known at design time, but not yet the complete specification. For tubes for example the tube diameter is not known at design time, since it depends on the bundle diameter of a specific configuration.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartVersion	alternativePartVersions	0..*		N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PartUsage		partSubstitution	0..1	N	

7.3.12 Class PartVersion

The PartVersion is one of the two anchors for PDM information in the VEC. All technical information about a PartVersion is contained in one or more documents. These describing elements are normally referencing to the PartVersion.

General Information

Base Classifier	ItemVersion
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

partNumber	String	1	The partNumber is the major identifier of a PartVersion. The format is user defined and respectively company specific. For all VEC-documents a PartVersion-instance can be trusted to be identical if the combination of partNumber, partVersion and companyName is identical.
partVersion	String	1	The partVersion specifies the version index of a part (see also partNumber).
primaryPartType	PrimaryPartType	1	The primary type of the part defines the type of the part (e.g. ConnectorHousing, Fixing, etc.) Since the VEC supports dual use parts (e.g. Fixing & WireProtection) the primary part type is necessary to define which specification associated to part is the primary character of the part. Therefore, all primary part types correspond to a PartOrUsageRelatedSpecification (e.g. ConnectorHousing --> ConnectorHousingSpecification).
isPreferredPart	String	0..1	Flags a part as "preferred" by the means of being a preferred part out of a group of parts with identical technical properties. The preferred part should be used, if the other properties of a couple of parts do not allow a distinct decision. (see KBLFRM-311)
preferredUseCase	String	0..1	Defines the function for which the part was initially designed. (e.g. "Grommet for Hatch", "...passenger compartment",...) This is an important information for searching and selecting parts in the context of KOMP.
aliasId	AliasIdentification	0..*	Room to specify additional identifiers for the PartVersion. This field must not be used for alternative PartNumbers. It is intended for identifiers others than PartNumbers, such as human readable identifiers printed on the part e.g. a number of fuse or a relay. Therefore, it does not have to be strictly unique.
nature	PartNature	0..1	The *nature* specifies how the *PartVersion* can be used in the different processes or the significance of the *PartVersion* in the process. The *nature* of a *PartVersion* is normally inherent and does not change. If the *nature* the underlying part changes a new (other) *partNumber* is assigned to the part and respectively a new *PartVersion* is created. It used to differentiate for example normal (productive) part numbers from preliminary (prototypic) part numbers.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Project	project	0..1	0..*	N	References the project that develops the PartVersion.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PartRelation	0..*	accessoryPart	1..*	N	References the PartVersions that are related by the PartRelation.
PartOrUsageRelatedSpecification	0..*	describedPart	0..*	N	References the PartVersion(s) to which the information defined in this specification applies. Example: If the PartOrUsageRelatedSpecification is a GeneralTechnicalPartSpecifcation and it defines that the color is "green" then all PartVersion referenced by this association are "green".

TerminalPairing	0..*	secondTerminal	1	N	References the second terminal of the TerminalPairing (first and second does not imply any specific order).
VecContent	1	partVersion	0..*	Y	Specifies the PartVersions contained in the VEC-file.
PartOccurrence	0..*	part	0..1	N	References the PartVersion that is instantiated by this PartOccurrence.
Mapping	0..*	A	1	N	
TerminalPairing	0..*	firstTerminal	1	N	References the first terminal of the TerminalPairing.
PartSubstitutionSpecification		alternativePartVersions	0..*	N	
SheetOrChapter	0..*	referencedPart	0..*	N	The association is an informative link which PartVersions are described by the SheetOrChapter.
UsageConstraintSpecification	0..*	constrainedParts	0..*	N	References the *PartVersions* to which this *UsageConstraintSpecification* applies.
DocumentVersion	0..*	referencedPart	0..*	N	The association is an informative link which PartVersions are described by the DocumentVersion.
Mapping	0..*	B	1	N	

7.3.13 Class RoutableElement

A RoutableElement is an element that can be routed, which mean it is possible to assign it to a Path in the Topology.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	true

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Routing	0..*	routedElement	1	N	Specifies the Element that is routed.

7.3.14 Class SheetOrChapter

Documents can be structured into sheets or chapters. Since it is possible, that one document describes a couple of parts it is necessary to be able to specify which sheet or chapter contains the description of which part. (see KBLFRM-308)

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

identification	String	1	The identification of the sheet or chapter within the document. This value must be distinct within the context of a document.
sheetNumber	String	0..1	The sheetNumber is the major identifier of a SheetOrChapter. The format is user defined and respectively company specific. This field has to be used if a SheetOrChapter has its own "DocumentNumber".
sheetVersion	String	0..1	The sheetVersion specifies the version index of a sheet (see also sheetNumber)
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the SheetOrChapter.
sheetFormat	String	0..1	Defines the format (e.g. size) of the SheetOrChapter.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Specification	specification	0..*	0..1	Y	Specifies the Specifications contained in the SheetOrChapter. All structured, technical information in the VEC is described by such Specifications.
ChangeDescription	changeDescription	0..*	0..1	Y	Specifies the change history of the SheetOrChapter.
PartVersion	referencedPart	0..*	0..*	N	The association is an informative link which PartVersions are described by the SheetOrChapter.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
DocumentVersion	1	sheetOrChapter	0..*	Y	Specifies SheetOrChapters defined in this DocumentVersion. These are especially useful if the DocumentVersion represents an external reference.
DocumentRelatedAssignmentGroup	0..*	relatedSheetOrChapter	0..1	N	Allows a more specific relationship to a *SheetOrChapter* within the *relatedDocumentVersion.*
DocumentBasedInstruction	0..*	referencedSheetOrChapter	0..1	N	References the SheetOrChapter that is used as an Instruction.

7.3.15 Class Specification

Abstract super-class for all specifications. Every technical information exchanged with the VEC is contained in the different specializations of a specification.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment

identification	String	1	Specifies a unique identification of the specification. The identification is guaranteed to be unique within the document containing the specification. For all VEC-documents a Specification-instance can be trusted to be identical if the DocumentVersion-instance is the same (see DocumentVersion) and the identification of the Specification is the same.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the specification.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
DocumentVersion	0..1	specification	0..*	Y	Specifies the Specifications contained in the DocumentVersion. All structured, technical information in the VEC is described by such Specifications.
SheetOrChapter	0..1	specification	0..*	Y	Specifies the Specifications contained in the SheetOrChapter. All structured, technical information in the VEC is described by such Specifications.

7.3.16 Class VecContent

The VecContent is the XML-Root node for any VEC-Document.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	xml-root
Is Abstract	false

Attributes

Name	Type	Mult	Comment
vecVersion	VecVersion	1	Specifies the version of the VEC used for the file.
generatingSystemName	String	0..1	Specifies the name of the system that has generated the VEC-file.
dateOfCreation	Date	0..1	Specifies the date of creation of the VEC-file.
generatingSystemVersion	String	0..1	Specifies the version of the system that has generated the VEC-file.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Unit	unit	0..*	1	Y	Specifies the Units used in the VEC-file.
ItemHistoryEntry	itemHistoryEntry	0..*	1	Y	Specifies the ItemVersionHistoryEntries for ItemVersions contained in the VEC-file.
PartVersion	partVersion	0..*	1	Y	Specifies the PartVersions contained in the VEC-file.
CopyrightInformation	standardCopyrightInformation	0..1	0..*	N	References the *CopyrightInformation* that is in effect for the complete content of this *VecContent*. It is applied to all *ItemVersions* that do not references their own individual *CopyrightInformation.*

Contract	contract	0..*	1	Y	Specifies the contracts used in the VEC-file.
Project	project	0..*	1	Y	Specifies the Projects used in the VEC-file.
CopyrightInformation	copyrightInformation	0..*	1	Y	Specifies the CopyrightInformation used in the VEC-file.
DocumentVersion	documentVersion	0..*	1	Y	Specifies the DocumentVersions contained in the VEC-file.

7.3.17 Enumeration AliasIdentificationType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
UUID		Defines that this AliasIdentification represents a "Universally Unique Identifier". Although a UUID is technical definition of a 128-Bit number, the primary relevance of this type is not its technical representation, but it's other properties. An AliasIdentification with the type "UUID" of an element is its unique identifier, that is constant over time, never changes and is never reused for other elements. Such an AliasIdentification can be used to trace elements through different systems, companies and processes.

7.3.18 Enumeration BaselineContent

Enumeration to define the state of the content of the baseline in regard of its defined scope.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Partial		A partial baseline does not contain all elements of the final scope of the baseline (e.g. it is a baseline created during the development process).
Complete		A complete baseline contains all elements of the final scope of the baseline.

7.3.19 Enumeration BaselineState

Enumeration the define the valid states of a baseline.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Draft		Draft means that the baseline is not finalized yet and new ItemVersions can be added without the necessity to create a new version of the baseline itself.
Frozen		Frozen means that the baseline is finalized and new ItemVersions must not be added without creating a new version of the baseline itself.

7.3.20 Enumeration DocumentType

Defines the predefined *DocumentTypes* of a *PartVersion.* A certain *DocumentType* has normally a typical set of information that is defined within its scope. E.g. a part master document contains *Specifications* that are used for the description of a defined *PartVersion.*

The content and the degree of information a DocumentType may vary in the different processes.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
BaselineDefinition		The *DocumentVersion* represents the definition of a baseline (see *BaselineSpecification*).
PartMaster		Documents of this type describe the master data of a part / component (e.g. a *ConnectorHousing*, a *Terminal*, a *Wire*).
NetworkArchitecture		
SystemSchematic		
WiringDescription		
HarnessDescription		Documents of this type contain the description of a *Harness* (which is for example the scope of the KBL).
InstallationDescription		
ManufacturingDescription		
RequirementsDescription		A requirements description is some kind of document that contains requirements for a certain part or a group of parts, their design or their properties. A requirements description can be a requirements specification (e.g. REQ-IF) or any other document containing requirements (e.g. a norm).
ChangeDescription		
GraphicsSymbol		Documents of this type represent a graphical symbol (2D), which is normally an external file (e.g. an SVG, PNG). The use case for this document type are symbols that are used in 2D harness and form board drawings.
GeometryModel		Documents of this type represent a geometry model (3D), which is normally an external file (e.g. a JT File or some native file of a 3D modelling tool). Typical use cases are for example the models of components (e.g. connectors) used in a 3D model or the 3D visualization of a Harness.
MasterDataDefinition		Documents of this type master data definition e.g. list valid *UsageNodes* or list valid *Signals*.
HarnessCoupling		*DocumentVersions* of this type define the coupling information of wiring harnesses in a vehicle network. The *DocumentVersion* contains the necessary *CouplingSpecifications* and the *PartUsages* for the coupling devices.
SchematicSymbol		Documents of this type represent a graphical symbol (2D), which is normally an external file (e.g. an SVG, PNG). The use case for this document type are symbols that are used in schematic diagrams.
NetworkSymbol		Documents of this type represent a graphical symbol (2D), which is normally an external file (e.g. an SVG, PNG). The use case for this document type are symbols that are used in network / architecture diagrams.

ComponentDrawing		*DocumentVersion* of this type represent the drawing of a component (Deutsch: Einzelteilezeichnung). Those *DocumentVersion* are normally used as external reference to the document containing the graphical representation of the component and do not contain *Specifications*. *DocumentVersion* describing a component in terms of the VEC (with *Specifications*) shall have the *DocumentType PartMaster.*
DeviationTable		Documents that define allowed deviations from specified part numbers.

7.3.21 Enumeration LanguageCode

Enumeration for the definition of ISO language codes. (see KBLFRM-317)

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Aa		Afar
Ab		Abkhazian
Af		
Am		
Ar		
As		
Ay		
Az		
Ba		
Be		
Bg		
Bh		
Bi		
Bn		
Bo		
Br		
Ca		
Co		

Cs		
Cy		
Da		
De		
Dz		
El		
En		
Eo		
Es		
Et		
Eu		
Fa		
Fi		
Fj		
Fo		
Fr		
Fy		
Ga		
Gd		
Gl		
Gn		
Gu		
Ha		
Hi		
He		
Hr		
Hu		

Hy		
Ia		
Id		
Ie		
Ik		
In		
Is		
It		
Iu		
Iw		
Ja		
Ji		
Jw		
Ka		
Kk		
Kl		
Km		
Kn		
Ko		
Ks		
Ku		
Ky		
La		
Ln		
Lo		
Lt		
Lv		

Mg		
Mi		
Mk		
Ml		
Mn		
Mo		
Mr		
Ms		
Mt		
My		
Na		
Ne		
Nl		
No		
Oc		
Om		
Or		
Pa		
Pl		
Ps		
Pt		
Qu		
Rm		
Rn		
Ro		
Ru		
Rw		

Sa		
Sd		
Sg		
Sh		
Si		
Sk		
Sl		
Sm		
Sn		
So		
Sq		
Sr		
Ss		
St		
Su		
Sv		
Sw		
Ta		
Te		
Tg		
Th		
Zi		
Tk		
Tl		
Tn		
To		
Tr		

Ts		
Tt		
Tw		
Ug		
Uk		
Ur		
Uz		
Vi		
Vo		
Wo		
Xh		
Yi		
Yo		
Za		
Zh		Chinese
Zu		Zulu

7.3.22 Enumeration LocalizedTypedStringType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Title		The LocalizedTypedString represents a Title in accordance to ISO 7200.
SupplementaryTitle		The LocalizedTypedString represents a SupplementaryTitle in accordance to ISO 7200.

7.3.23 Enumeration PartNature

OpenEnumeration that defines the nature of a *PartVersion*. The nature specifies how the *PartVersion* can be used in the different processes or the significance of the *PartVersion* in the process.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Preliminary		*Preliminary PartVersion* represent part numbers which are used as place holders for parts that are not yet developed, or which are representing prototype parts that are not allowed in serial production.
Productive		*Productive PartVersions* represent regular part numbers that are used in serial production.
CustomerOrder		*CustomerOrder PartVersions* represent part numbers that are not regularly used in serial production. They are only used to fulfil special customer orders (e.g. an adapter connector for equipment of a special purpose vehicle).

7.3.24 Enumeration PrimaryPartType

The primary type of the part defines the type of the part (e.g. ConnectorHousing, Fixing, etc.) Since the VEC supports dual use parts (e.g. Fixing & WireProtection) the primary part type is necessary to define which specification associated to part is the primary character of the part. Therefore, all primary part types correspond to a PartOrUsageRelatedSpecification (e.g. ConnectorHousing --> ConnectorHousingSpecification).

The primary part type 'Other' is used if the PartVersion is not further specified by the VEC, which means it has no PartOrUsageRelatedSpecification, only a GeneralTechnicalPartSpecification or a direct instance of PartOrUsageRelatedSpecification.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		Comment
Antenna		
Battery		
BoltMountedFixing		
BoltTerminal		
CableDuct		
CableTie		
Capacitor		
CavityAccessory		
CavityPlug		
CavitySeal		
ConnectorHousing		
ConnectorHousingCap		
CorrugatedPipe		

Diode		
EdgeMountedFixing		
EEComponent		
Ferrite		
Fitting		
Fixing		
Fuse		
Grommet		
HoleMountedFixing		
MultiCavityPlug		
MultiCavitySeal		
MultiFuse		
Other		
OpenWireEnd		
PartStructure		
PluggableTerminal		
PotentialDistributor		
Relay		
RingTerminal		
ShrinkableTube		
SpliceTerminal		
Stripe		
Tape		
Terminal		
Tube		
Wire		
WireEndAccessory		

WireProtection		

7.3.25 Enumeration VecVersion

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		Comment
1.1.1		
1.1.2		
1.1.3		
1.2.0		

7.4 Module coupling

7.4.1 Class CavityCoupling

A *CavityCoupling* defines the mapping between two cavities of the *ConnectorHousingRoles* associated with the *CouplingPoint*.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavityReference	secondCavity	1	0..*	N	
CavityReference	firstCavity	1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SlotCoupling	1	cavityCoupling	0..*	Y	

7.4.2 Class CouplingPoint

A *CouplingPoint* defines a single coupling. If a coupling takes place, all sub elements are connected. If the coupling is disconnected, all subelements are disconnected.

If a coupling occurs between two connectors, and not just between two terminals, the *CouplingPoint* references the respective *ConnectorHousingRoles*.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConnectorHousingRole	secondConnector	0..1	0..*	N	
ConnectorHousingRole	firstConnector	0..1	0..*	N	
SlotCoupling	slotCoupling	0..*	1	Y	
MatingPoint	matingPoint	0..*	1	Y	Specifies the MatingPoints defined by the MatingSpecification.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CouplingSpecification	1		0..*	Y	

7.4.3 Class CouplingSpecification

Specification for the description of a Coupling. A coupling allows the mapping between independent harness sections or EEComponents of the electrical system. This is done by the mapping of either-or both, ConnectorHousingRoles & TerminalRoles of one side to the other.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

CouplingPoint		0..*	1	Y	

7.4.4 Class MatingDetail

If the mating of the two terminals is not unambiguously, a MatingDetail can specify the TerminalReceptions that are mated.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TerminalReception Reference	secondTerminalReception	1	0..*	N	References the second terminal reception that is mated.
Connection	connection	0..1	0..*	N	References the *Connection* that is realized by this *MatingPointDetail.* For example, when a connection is realized by directly plugging or screwing two E/E components together. The definition at level of the *MatingDetail* might be required if the *TerminalRole* of the MatingPoint carries multiple different potentials (e.g. Coax).
TerminalReception Reference	firstTerminalReception	1	0..*	N	References the first terminal reception that is mated.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
MatingPoint	1	matingDetail	0..*	Y	Specifies the MatingDetails, if a detailed description of the relationships between TerminalReceptions and TerminalReceptions is needed.

7.4.5 Class MatingPoint

A MatingPoint defines the Mating of two terminals. This normally occurs when two inliners are connected. Then terminals of one side (female) are mated with terminals of the other side (male).

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the MatingPoint. The identification is guaranteed to be unique within the MatingSpecification.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TerminalRole	firstTerminalRole	1	0..*	N	References the first terminal that is mated.
Connection	connection	0..1	0..*	N	References the *Connection* that is realized by this *MatingPoint.* For example, when a connection is realized by directly plugging or screwing two E/E components together.
MatingDetail	matingDetail	0..*	1	Y	Specifies the MatingDetails, if a detailed description of the relationships between TerminalReceptions and TerminalReceptions is needed.
TerminalRole	secondTerminalRole	1	0..*	N	References the second terminal that is mated.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CouplingPoint	1	matingPoint	0..*	Y	Specifies the MatingPoints defined by the MatingSpecification.

7.4.6 Class SlotCoupling

A *SlotCoupling* defines the mapping between two slots of the *ConnectorHousingRoles* associated with the *CouplingPoint*.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
AbstractSlotReference	secondSlot	1	0..*	N	
AbstractSlotReference	firstSlot	1	0..*	N	
CavityCoupling	cavityCoupling	0..*	1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CouplingPoint	1	slotCoupling	0..*	Y	

7.5 Module custom_properties

7.5.1 Class BooleanValueProperty

A custom property with a boolean value.

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	Boolean	1	Defines the value of the CustomProperty.

7.5.2 Class ComplexProperty

A custom property that represents a tuple of values.

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CustomProperty	customProperty	0..*	1	Y	The customProperties that represent the individual values of the complex property.

7.5.3 Class CustomProperty

Abstract base class for custom properties. Basically, a custom property is key / value pair. The key (propertyType) defines the meaning of the value. A custom property can either be a simple value (string), a numerical value or a value range.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
propertyType	String	1	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ExtendableElement	1	customProperty	0..*	Y	Specifies the CustomProperties of the ExtendableElement.
ComplexProperty	1	customProperty	0..*	Y	The customProperties that represent the individual values of the complex property.

7.5.4 Class DateValueProperty

A custom property with a date value.

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	Date	1	Defines the value of the CustomProperty.

7.5.5 Class DoubleValueProperty

A custom property with a double value.

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	Double	1	Defines the value of the CustomProperty.

7.5.6 Class IntegerValueProperty

A custom property with an integer value.

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	Integer	1	Defines the value of the CustomProperty.

7.5.7 Class LocalizedStringProperty

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	LocalizedString	1	Defines the value of the *CustomProperty*.

7.5.8 Class NumericalValueProperty

A custom property with a numerical value. (see KBLFRM-319)

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	NumericalValue	1	Defines the value of the *CustomProperty*.

7.5.9 Class SimpleValueProperty

A custom property with a simple value (string).

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	String	1	Defines the value of the *CustomProperty*.

7.5.10 Class ValueRangeProperty

A custom property with a value range. (see KBLFRM-319)

General Information

Base Classifier	CustomProperty
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
value	ValueRange	1	Defines the value of the CustomProperty.

7.6 Module electrical_parts

7.6.1 Class AbstractSlot

An *AbstractSlot* is a geometrical place in a connector housing, which can contain / group cavities. This can be either direct, if it is *Slot* and indirect if it is a *ModularSlot*.

If it is a *Slot*, then it is an inseparable part of the connector housing, which means it is created during the manufacturing process of the connector housing.

If it is a *ModularSlot* it is a place where one or more other connector housing can be place during the assembly.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
slotNumber	String	0..1	Specifies the number of the slot. This must be unique within a ConnectorHousingSpecification.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Coding	coding	0..1	0..1	Y	Defines coding of the slot that is satisfied by the Slot.
SlotSpecification	slotSpecification	0..1	0..*	N	References the SlotSpecification that is satisfied by the slot.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConnectorHousing Specification	1	slot	0..*	Y	Specifies the slots forming the ConnectorHousing.
AbstractSlotReference	0..*	referencedSlot	1	N	Points to the slot referenced by the slot reference.

7.6.2 Class AntennaSpecification

Specification of the electrological aspects of an antenna.

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
fMin	NumericalValue	0..1	Specifies the minimum operating frequency of the antenna.
fMax	NumericalValue	0..1	Specifies the maximum operating frequency of the antenna.
impedance	NumericalValue	0..1	Specifies the impedance of the antenna.

7.6.3 Class BatterySpecification

Specification of the electrological aspects of a battery.

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
u	NumericalValue	0..1	Specifies the nominal voltage of the battery.
i	NumericalValue	0..1	Specifies the current the battery provides.
iCool	NumericalValue	0..1	Specifies the battery's current in cool state.
capacity	NumericalValue	0..1	Specifies the power capacity of the battery.

7.6.4 Class BoltTerminalSpecification

Specification for the definition of bolt terminals. These are the counterparts to ring terminals.

General Information

Base Classifier	TerminalSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
boltDiameter	NumericalValue	0..1	Specifies the diameter of the bolt in a nominal way.
boltHeight	NumericalValue	0..1	Specifies the height of the bolt (the height of the part to which ring terminals can be attached).

boltNominalSize	String	0..1	Defines the size (diameter) of the bolt in a nominal way (e.g. "M8").
boltType	String	0..1	Specifies the type of the bolt.
torsionProtection	Boolean	0..1	Specifies if the bolt provides torsion protected or not.
maxTerminalCount	Integer	0..1	Defines the maximum number of (ring) terminals that can be attached to this bolt at the same time.

7.6.5 Class BridgeTerminalSpecification

A bridge terminal is a part that behaves like terminal but has no *WireReceptions.* It is used to create short circuit between different pins in a connector. In its use, it can realize a schematic connection on its own and independently of other components.

General Information

Base Classifier	TerminalSpecification
Applied Stereotype	
Is Abstract	false

7.6.6 Class CapacitorSpecification

Specification of the electrological aspects of a capacitor.

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
capacity	NumericalValue	0..1	
impedance	NumericalValue	0..1	
uMax	NumericalValue	0..1	The breakdown voltage of the capacitor.

7.6.7 Class Cavity

A cavity is a defined space in a connector housing for location of an electrical terminal or cavity plug or seal. A cavity may also be empty.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

available	Boolean	0..1	Defines whether the cavity is available for contacting. If the cavity is not available, it means that it is completely closed.
cavityNumber	String	0..1	Provides an identifier for the cavity. The cavity number needs to be unique within a *Slot*.
hasIntegratedTerminal	Boolean	0..1	Defines whether the cavity has an integrated terminal (for example an IDC cavity) or if an additional terminal is required. If this attribute is "true", the cavity can reference a *TerminalSpecification* as *integratedTerminalSpecification* in order to specify the integrated terminal.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TerminalSpecification	integratedTerminalSpecification	0..1		N	Specifies the terminal, if the cavity has an integrated terminal (e.g. an IDC).
CavitySpecification	cavitySpecification	0..1	0..*	N	References the CavitySpecification that is satisfied by the cavity.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CavityAddOn	0..*	cavity	1	N	
SegmentConnectionPoint	0..*	reachableCavities	0..*	N	Specifies the *Cavities* that are reachable with wires through this *SegmentConnectionPoint.*
Slot	1	cavity	1..*	Y	Specifies the Cavities forming the Slot.
PinComponent	0..*	referencedCavity	0..1	N	Defines the cavity in the corresponding ConnectorHousingSpecification of the HousingComponent where the PinComponent is located. (see KBLFRM-300)
SealedCavitiesAssignment	0..*	sealedCavities	1..*	N	Specifies the Cavities that are sealed.
OpenCavitiesAssignment	0..*	openCavities	1..*	N	Specifies the cavities that are open.
CavityReference	0..*	referencedCavity	1	N	Points to the cavity referenced by the cavity reference.

7.6.8 Class CavityAccessorySpecification

A *CavityAccessory* is a non-electrical part used in a cavity with no sealing properties (e.g. a wire fixation).

General Information

Base Classifier	CavityPartSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

wireElementOutsideDiameter	ValueRange	0..1	Specifies a range of valid wire diameters to which the cavity accessory fits.
wireReceptionType	WireReceptionType	0..1	Specifies the wireReceptionType to which the cavity accessory fits.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CavityAccessoryRole	0..*	cavityAccessorySpecification	1	N	

7.6.9 Class CavityAddOn

Specifies the wire addon needed to reach a *Cavity* from a specific *SegmentConnectionPoint*. For each *SegmentConnectionPoint* there shall be no more than one add-on value per cavity and type.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
wireAddOn	NumericalValue	1	Specifies the wire length add on needed for the cavity.
type	WireAddOnType	1	Defines the type of the add-on (see *CavityAddOn*).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Cavity	cavity	1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SegmentConnectionPoint	1	cavityAddOns	0..*	Y	

7.6.10 Class CavityPartSpecification

The *CavityPartSpecification* is an abstract class for common properties of non-electrical parts that are used in *Cavities*.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
cavityDimension	Size	0..*	Specifies a valid cavity dimensions to which the cavity part fits. The dimension defines the size of the sealing area of the cavity (crimp end), not in the contacting area (box end). Note: CavityDimension is of type Size which is defined as x & y with type NumericalValue. NumericalValue can define tolerances. So, a cavity dimension is not necessarily a single fixed value.
hardness	NumericalValue	0..1	Specifies the hardness of the cavity seal.
geometry	SealingGeometry	0..1	Defines the geometry of the cavity sealing. This attribute is defined as an OpenEnumeration.
compatibleTerminalType	TerminalType	0..*	Defines a list of terminal types that are compatible to this CavitySealSpecification. This defines as well the compatible cavities, since a plug is normally used when no terminals are present.
compatibleCavityGeometry	CavityGeometry	0..*	Defines a list of *CavityGeometries* that are compatible with this cavity part.

7.6.11 Class CavityPlugSpecification

Specification for the definition of cavity seals. A cavity plug is a watertight non-electrical object to fill an empty cavity.

General Information

Base Classifier	CavityPartSpecification
Applied Stereotype	
Is Abstract	false

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CavityPlugRole	0..*	cavityPlugSpecification	1	N	References the *CavityPlugSpecification* that is instanced by this *CavityPlugRole.*

7.6.12 Class CavitySealSpecification

Specification for the definition of cavity seals. A *CavitySeal* is a watertight non-electrical object to fill a populated Cavity.

General Information

Base Classifier	CavityPartSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

wireElementOutsideDiameter	ValueRange	0..1	Specifies a range of valid wire diameters to which the cavity seal fits.
wireReceptionType	WireReceptionType	0..1	Specifies the wireReceptionType to which the cavity seal fits.
insideDiameter	NumericalValue	0..1	Defines the inside diameter in the relaxed state for a cavity seal.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CavitySealRole	0..*	cavitySealSpecification	1	N	References the *CavitySealSpecification* that is instanced by this *CavitySealRole.*

7.6.13 Class CavitySpecification

Specification for the definition of cavities.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
angle	NumericalValue	0..2	Specifies the angle against two planes of the connector housing a terminal used in this cavity can be buckled.
cavityDesign	String	0..1	Deprecated (since Version 1.1.4): This attribute has been marked as deprecated, as it has been replaced by the more meaningful mechanism with *TerminalTypes.*
geometry	CavityGeometry	0..1	Defines the geometry of a cavity in the sealing area (crimp end).
cavityDimension	Size	0..1	Specifies the dimension of the cavity in the sealing area of the cavity (crimp end), not in the contacting area (box end).
minWireElementOutsideDiameter	NumericalValue	0..1	Specifies the minimum diameter a wire is allowed to have to fit into the cavity. This definition is necessary, since wires that are too small might cause movements and in acceptable torsion forces on terminals.
maxWireElementOutsideDiameter	NumericalValue	0..1	Specifies the maximum diameter a wire is allowed to have to fit into the cavity.
primaryLockingType	PrimaryLockingType	0..1	Specifies if the cavity has a primary locking and of what type it is.
sealable	Boolean	0..1	Specifies if the cavity is sealable.
compatibleTerminalType	TerminalType	0..*	Defines a list of terminal types that are compatible to this CavitySpecification.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

Cavity	0..*	cavitySpecification	0..1	N	References the CavitySpecification that is satisfied by the cavity.

7.6.14 Class Coding

Specifies the coding of a slot or a connector housing.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
coding	CodingName	1	Specifies the name of the coding.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
AbstractSlot	0..1	coding	0..1	Y	Defines coding of the slot that is satisfied by the Slot.
ConnectorHousing Specification	0..1	coding	0..1	Y	Defines coding of the connector housing that is satisfied by the connector housing.

7.6.15 Class ConductorCurrentInformation

The *ConductorCurrentInformation* specifies the maximum current for which a conductor is approved. As the maximum current is dependent from the voltage and the environment temperature it is modelled as a class and not only as an attribute.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
maxCurrent	NumericalValue	1	The maximum current value.
environmentTemperature	NumericalValue	1	The environment temperature for which this maximum current value is applicable.
voltage	NumericalValue	1	The voltage for which this maximum current value is applicable.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

ConductorSpecification	1	currentInformation	0..*	Y	Specifies the current information of the conductor. These are the maximum currents for which the conductor is approved.

7.6.16 Class ConductorMaterial

ConductorMaterial is a helper class to specify *validConductorMaterials.* This is necessary, since all attributes of the type *Material* have a multiplicity of * with the semantics that it always defines one *Material* with the possibility to define it in different reference systems.

Since a wire reception can have more than one *validConductorMaterial* this container class is necessary to keep the semantics clear.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
material	Material	1..*	Specifies the material.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireReceptionSpecification	1	validConductorMaterials	0..*	Y	Specifies the materials of a conductor, that are valid to use with this *WireReceptionSpecification*. This material shall be matched against the *ConductorSpecification.material.*

7.6.17 Class ConductorSpecification

Specification for the definition of conducting properties of a WireElement.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
crossSectionArea	NumericalValue	0..1	Specifies the cross-section area of the conductor (e.g. 0,5 mm²). The cross-section area is a nominal value, which refers to the conducting properties of the conductor normalized to the properties of a full material core.
massInformation	MassInformation	0..1	Specifies the mass information of the conductor. In most cases this mass information is known as copper weight and is normally specified as mass per length (e.g. gram per meter).
material	Material	0..*	Specifies the material of the conductor.

resistance	NumericalValue	0..1	Specifies the electrical resistance of the conductor. In most cases this is specified as resistance per length (e.g. Ohm per meter).
structure	String	0..1	Specifies the structure of the conductor (e.g. symmetrical, asymmetrical).
type	String	0..1	Specifies the type of the conductor.
numberOfStrands	NumericalValue	0..1	Specifies the number of strands in one conductor. If the conductor is solid than the number of strands would be one.
platingMaterial	Material	0..*	Specifies the plating material of the conductor.
strandDiameter	NumericalValue	0..1	Specifies the diameter of a single strand in the conductor.
voltageRange	NumericalValue	0..1	Specifies the voltageRange for which the conductor is approved.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConductorCurrentInformation	currentInformation	0..*	1	Y	Specifies the current information of the conductor. These are the maximum currents for which the conductor is approved.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TerminalPairing	0..*	referencedCoreSpecification	1	N	References the CoreSpecification that is used on both sides of the ContactSystem.
WireElementSpecification	0..*	conductorSpecification	0..1	N	If the *WireElement* has a core then the specification of the core is referenced here.
InternalComponentConnection	0..*	conductorSpecification	0..1	N	
Signal		recommendedConductorSpecification	0..1	N	Defines a recommended Specification for the cores that implement this signal.

7.6.18 Class ConnectorHousingCapSpecification

Specification for the definition of caps (backshells) of connectors. Different caps can add additional wire length addons to a connector housing.

A 'cap' which already defines the number of cavities, coding etc. is in the VEC defined by *ConnectorHousingSpecification* with *ModularSlots* and not by a *ConnectorHousingCapSpecification*.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

wireAddOn	NumericalValue	0..1	Specifies the wire length add on needed for the cap.
outletDirection	ConnectorOutletDirection	0..1	Defines the *OutletDirection* for wires. This attribute is defined as an *OpenEnumeration*.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConnectorHousingCapRole	0..*	connectorHousingCapSpecification	1	N	References the *ConnectorHousingCapSpecification* that is instanced by this *ConnectorHousingCapRole*.

7.6.19 Class ConnectorHousingSpecification

Specification for the definition of connector housings. A connector housing consists of a one or more slots. In the means of the VEC, a connector housing can be a conventional connector housing, a contact module or a connector shell.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
averageWireLengthAddOn	NumericalValue	0..1	Specifies the average wire length add on for this connector.
voltageRange	ValueRange	0..1	Specifies the allowed voltage range for the connector housing.
coupleable	Boolean	0..1	Defines whether the connector is coupleable or not. Connectors that are coupleable can be used in an inline position. Connectors that are not coupleable can be connected only to an ECU or something similar.
connectorPositionAssurance	Boolean	0..1	If *true* the *ConnectorHousing* has a connector position assurance (CPA). A CPA is some sort of feature of a connector, that secures the connector in its correctly assembled position with its mating part.
outletDirection	ConnectorOutletDirection	0..1	Defines the *OutletDirection* for wires. This attribute is defined as an *OpenEnumeration*.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
SegmentConnectionPoint	segmentConnectionPoint	0..*	1	Y	Specifies the *SegmentConnectionPoints* the connector housing.
Coding	coding	0..1	0..1	Y	Defines coding of the connector housing that is satisfied by the connector housing.
AbstractSlot	slot	0..*	1	Y	Specifies the slots forming the ConnectorHousing.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConnectorHousingRole	0..*	connectorHousingSpecification	1	N	References the *ConnectorHousingSpecification* that is instanced by this *ConnectorHousingRole.*
HousingComponent	0..*	housingSpecification	0..1	N	References the ConnectorHousingSpecification that is describing the connector interface of the HousingComponent (e.g. Slots, Cavities, Design, Coding).

7.6.20 Class CoreSpecification

Defines the properties of a circular conductor (core) which are specific for them.

General Information

Base Classifier	ConductorSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
outsideDiameter	NumericalValue	0..1	Specifies the outside diameter of the core.

7.6.21 Class DiodeSpecification

Specification of the electrological aspects of a diode.

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
thresholdVoltage	NumericalValue	0..1	Voltage at which the diode starts conducting in forward direction.
breakDownVoltage	NumericalValue	0..1	Voltage at which the diode starts conducting in reverse direction.
iMax	NumericalValue	0..1	Specifies the maximum electric current tolerated by the diode.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinComponent	anode	0..1	0..*	N	
PinComponent	cathode	0..1	0..*	N	

7.6.22 Class EEComponentSpecification

Base-class for the specification of electrological components, which are connected to the harness. Usually electrological components are not part of the harness e.g. a fuse, a switch or a control device. All EEComponents can have one or more HousingComponents which are possible interfaces for the connection to a harness.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
powerConsumption	PowerConsumption	0..*	Specifies the *PowerConsumptions* of this *EEComponentSpecification.*

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ExtensionSlot	extensionSlots	0..*	1	Y	Specifies the available ExtensionSlots of the EEComponent.
SwitchingState	states	0..*	1	Y	Specifies the available *SwitchingStates* of the EEComponent.
HousingComponent	housingComponent	0..*	1	Y	Specifies the available connector interfaces of the EEComponent.
InternalComponentConnection	connections	0..*	1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
EEComponentRole	0..*	EEComponentSpecification	1	N	References the *EEComponentSpecification* that is instanced by this *EEComponentRole.*

7.6.23 Class ExtensionSlot

An *ExtensionSlot* defines a slot within an EE-Component where other EE-Components can be plugged into (modular extension). This is necessary for example for modular power distributions.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Identification of the ExtensionSlot, which must be distinct for all ExtensionSlots of an EEComponent.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartRelation	allowedInserts	0..*	0..*	N	References the *PartRelations* that are valid inserts for this *ExtensionSlot.* This reference points to *PartRelations* in order to allow referencing indirectly a *PartVersion* if the description of individual *PartVersions* is done with one physical VEC file per *PartVersion* and to allow the expression of optional inserts, choices etc. However, inserts for an *ExtensionSlot* are always *EEComponents* by itself. Therefore, the referenced *PartVersion* shall have a *PrimaryPartType* = *EEComponent.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
EEComponentSpecification	1	extensionSlots	0..*	Y	Specifies the available ExtensionSlots of the EEComponent.
ExtensionSlotReference		extensionSlot	1	N	

7.6.24 Class FillerSpecification

Specification for the definition of filler elements in the wire.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
material	Material	0..*	Specifies the material of the filler.
diameter	NumericalValue	0..1	Specifies the diameter of the filler.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireElementSpecification	0..*	fillerSpecification	0..1	N	If the *WireElement* is a filler then the specification of the filler is referenced here.

7.6.25 Class FlatCoreSpecification

Defines the properties of a flat (rectangular) conductor which are specific for them.

General Information

Base Classifier	ConductorSpecification

Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
size	Size	0..1	Defines the size (width & height) of the flat core.

7.6.26 Class FuseComponent

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Identification of the FuseComponent, which must be distinct for all FuseComponents of an MultiFuseSpecification.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinComponent	connectedPins	0..2	0..*	N	
FuseSpecification	fuseSpecification	0..1		N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
MultiFuseSpecification	1	fuseComponents	0..*	Y	

7.6.27 Class FuseSpecification

Specification of the electrological aspects of a fuse.

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
fuseType	FuseType	0..1	The geometric type of a fuse.

iMax	NumericalValue	0..1	Specifies the maximum electric current tolerated by the fuse.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
FuseComponent		fuseSpecification	0..1	N	

7.6.28 Class GeneralTechnicalPartSpecification

Specification for the definition of common properties for technical parts.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
colorInformation	Color	0..*	Specifies the color of the part.
massInformation	MassInformation	0..*	Specifies the mass of the part.
materialInformation	Material	0..*	Specifies the material of a part.
robustnessProperties	RobustnessProperties	0..*	Specifies the robustness of a part.
temperatureInformation	TemperatureInformation	0..*	Specifies valid temperatures for a part.
fitRate	Double	0..1	The Failures In Time (FIT) rate of a device is the number of failures that can be expected in one billion (109) device-hours of operation.[14] (E.g. 1000 devices for 1 million hours, or 1 million devices for 1000 hours each, or some other combination.) (see https://en.wikipedia.org/wiki/Failure_rate#Units)
unspecifiedAccessoryPermitted	Boolean	0..1	Defines whether accessories which are not explicitly defined by a *PartRelation* may be used with instances of this part. If this attribute is not specified the default value is *true.*

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
BoundingBox	boundingBox	0..1		Y	Defines the bounding box of the part.
PartRelation	partRelation	0..*	1	Y	Specifies possible relations (accessories) of the specified part with other PartVersion (e.g. caps, clips).

7.6.29 Class HousingComponent

A HousingComponent describes the interface of an EEComponent with which it can be connected to another EEComponent or a harness. The characteristics of the interface can be described with a referenced ConnectorHousingSpecification. (see KBLFRM-300)

General Information

Base Classifier	ConfigurableElement ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Identification of the HousingComponent, which must be distinct for all HousingComponents of an EEComponent.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the *HousingComponent*.
compatibleTypes	HousingComponentType	0..*	The values of this attribute define the *HousingComponentTypes* that are valid to be associated with this *HousingComponent.* In other word, if this *HousingComponent* can be associated with a relay, a fuse, a connector housing of a harness. The values are matching the *PrimaryPartType* of the *PartVersion* of the component that should be associated (plugged) into this housing component.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinComponent	pinComponent	0..*	1	Y	Specifies the PinComponents of HousingComponent. (see KBLFRM-300)
ConnectorHousingSpecification	housingSpecification	0..1	0..*	N	References the ConnectorHousingSpecification that is describing the connector interface of the HousingComponent (e.g. Slots, Cavities, Design, Coding).

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
HousingComponentReference	0..*	housingComponent	1	N	Points to the HousingComponent referenced by the HousingComponent reference. (KBLFRM-401)
EEComponentSpecification	1	housingComponent	0..*	Y	Specifies the available connector interfaces of the EEComponent.

7.6.30 Class InsulationSpecification

Specification for the definition of insulation properties of a WireElement.

General Information

Base Classifier	Specification
Applied Stereotype	

Is Abstract	false

Attributes

Name	Type	Mult	Comment
baseColor	Color	0..*	Specifies the base color of the insulation.
firstIdentificationColor	Color	0..*	Specifies the first identification color of the insulation.
secondIdentificationColor	Color	0..*	Specifies the second identification color of the insulation.
labelIdentificationColor	Color	0..*	Specifies the color of a label printed on the insulation of the wire.
labelIdentificationType	String	0..1	Specifies the type of a label printed on the insulation of the wire (e.g. alpha numerical, bar code).
labelIdentificationValue	String	0..1	Specifies the value of a label printed on the insulation of the wire.
material	Material	0..*	Specifies the material of the insulation.
thickness	NumericalValue	0..1	Specifies the thickness of the insulation.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Signal		recommendedInsulationSpecification	0..1	N	Defines a recommended Specification for the insulation (e.g. the color) that implements this signal.
WireElementSpecification	0..*	insulationSpecification	0..1	N	If the *WireElement* has an insulation then the specification of the insulation is referenced here.

7.6.31 Class InternalComponentConnection

An *InternalComponentConnection* defines a conductive connection between *PinComponents* within an *EEComponent*. Such a connection can be statically permanent or dynamically switch (e.g. by a relais). This behaviour can be defined with a *SwitchingState*.

The electrical behaviour of an *InternalComponentConnection* can be further specified by a referenced *ConductorSpecification.*

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Identification of the *InternalComponentConnection*, which must be distinct for all *InternalComponentConnection* of an *EEComponent*.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

ConductorSpecification	conductorSpecification	0..1	0..*	N	
PinComponent	pins	2..*	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SwitchingState	0..*	switchedConnections	0..*	N	Specifies the *InternalComponentConnections* that are switched by this *SwitchingState*.
EEComponentSpecification	1	connections	0..*	Y	

7.6.32 Class InternalTerminalConnection

An InternalTerminalConnection represents an electrical connection within a terminal. For standard terminals all receptions (wire- and terminal-receptions) have an electrical connection. For non-standard terminals (e.g. coax) only some receptions have an electrical connection. The InternalTerminalConnection is modelled as a separate class and not as relationship between wire- and terminal-reception, since it is possible that a terminal has only one kind of reception (e.g. a parallel connector, a cavity bridge). (see KBLFRM-302)

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the InternalTerminalConnection. The identification is guaranteed to be unique within the TerminalSpecification (this might be for example an internal connection number).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireReception	wireReception	0..*	0..1	N	References the WireReceptions that participate in the InternalTerminalConnection.
TerminalReception	terminalReception	0..*	0..1	N	References the TerminalReceptions that participate in the InternalTerminalConnection.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TerminalSpecification	1	internalTerminalConnection	0..*	Y	Specifies the InternalTerminalConnections of the terminal.

7.6.33 Class ModularSlot

A ModularSlot is a place in a connector housing where different other connector housings can be placed during the assembly (e.g. by clicking them into the connector housing).

General Information

Base Classifier	AbstractSlot
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
optional	Boolean	0..1	Specifies whether the allocation of the ModularSlot is optional or not (can a usage of the connector housing leave this ModularSlot empty.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartRelation	allowedInserts	0..*	0..*	N	References the *PartRelations* that are valid inserts for this *ModularSlot.* This reference points to *PartRelations* in order to allow referencing indirectly a *PartVersion* if the description of individual *PartVersions* is done with one physical VEC file per *PartVersion* and to allow the expression of optional inserts, choices etc. However, inserts for a *ModularSlot* are always ConnectorHousings by itself. Therefore, the referenced *PartVersion* shall have a *PrimaryPartType = ConnectorHousing*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ModularSlotAddOn	0..*	slot	1	N	

7.6.34 Class ModularSlotAddOn

Specifies the wire addon needed to reach a *ModularSlot* from a specific *SegmentConnectionPoint*. The addon needed to reach the cavities of the insert(s) from this point is defined by the *ConnectorHousingSpecification* of the insert.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
wireAddOn	NumericalValue	1	Specifies the wire length add on needed for the modular slot.

Outgoing Relations

Other End	This	General

Type	Role	Mult	Mult	Agg	Comment
ModularSlot	slot	1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SegmentConnectionPoint	1	ModularSlotAddOns	0..*	Y	

7.6.35 Class MultiCavityPlugSpecification

Specification for the definition of cavity plugs that apply to more than one cavity. A cavity plug is a watertight non-electrical object to fill an empty cavity. MultiCavityPlugs are formed to fit into one connector / slot and to seal more than one cavity at once. Normally there are different variants of these MultiCavityPlugs that can seal a connector in different pinning scenarios. The cavities that are plugged by a MultiCavityPlug are defined with a SealedCavitiesAssignment.

General Information

Base Classifier	CavityPlugSpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
SealedCavitiesAssignment	assignment	0..*	1	Y	References the *SealedCavitiesAssignments* that are valid for this *MultiCavityPlug*. One individual *SealedCavitiesAssignment* is used for each connector housing that matches witch this *MultiCavityPlug.*

7.6.36 Class MultiCavitySealSpecification

Specification for cavity seals that apply to more than one cavity. A CavitySeal is a watertight non-electrical object to fill a populated Cavity. MultiCavitySeals are formed to fit into one connector / slot and to seal more than one cavity at once.

There are existing two types of MultiCavitySeals:

- In first type the MultiCavitySeal has an opening for all cavities of the connector. Each opening can be filled either with a wire (without an individual seal) or with a CavityPlug (e.g. a synthetic pin) or a MultiCavityPlug.
- In the second type, the MultiCavitySeal has a specific configuration of openings for some cavities of the connector. These MultiCavitySeals are sealing all cavities with an opening and a wire in it and all cavities where no opening in the MultiCavitySeal exists. For each opening that has no wire it an additional CavityPlug is needed.

The cavities that are left open by a MultiCavitySeal are defined with a SealedCavitiesAssignment.

General Information

Base Classifier	CavitySealSpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
OpenCavitiesAssignment	assignment	0..*	1	Y	References the *OpenCavitiesAssignments* that are valid for this *MultiCavitySeal*. One individual *OpenCavitiesAssignment* is used for each connector housing that matches witch this *MultiCavitySeal.*

7.6.37 Class MultiFuseSpecification

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
fuseType	String	0..1	The type of a fuse.
iMaxTotal	NumericalValue	0..1	Specifies the maximum electric current tolerated by the multifuse in total.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
FuseComponent	fuseComponents	0..*	1	Y	

7.6.38 Class OpenCavitiesAssignment

An OpenCavitiesAssignment groups the cavities of ONE connector that are open in a MultiCavitySeal. If a MultiCavitySeal fits into more than one connector, than there are as many OpenCavitiesAssignments.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Cavity	openCavities	1..*	0..*	N	Specifies the cavities that are open.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
MultiCavitySealSpecification	1	assignment	0..*	Y	References the *OpenCavitiesAssignments* that are valid for this *MultiCavitySeal*. One individual

					OpenCavitiesAssignment is used for each connector housing that matches witch this *MultiCavitySeal.*

7.6.39 Class OpenWireEndTerminalSpecification

Specification for the definition of open wire ends. An open wire end is an end of wire located at a specific location in the harness. Open wire ends are not connected to some other component, in contrast to regular connectors or ring terminals. Only one wire is ending at a single open wire end, otherwise it would be a splice.

General Information

Base Classifier	TerminalSpecification
Applied Stereotype	
Is Abstract	false

7.6.40 Class PartRelation

A *PartRelation* defines additional parts (e.g. accessories) for a specific part. These parts are in some way or usage scenario required for the part itself to be used. However, they are not included with the part number and have to ordered separately. This can be used for example for caps, cable ties etc.

The associated *PartRelations* of a *GeneralTechnicalPartSpecification* represent a configurable bill of material that can/must be ordered together with the part, when it is used. Each *PartRelation* represents an item / line in this bill of material. The semantic by which a *PartRelation* is resolved to *PartVersions* is defined by the *PartRelationType.* If multiple *PartRelations* resolve to the same *PartVersions* the resulting bill of material is the sum of them.

If a *PartRelation* references more than one *accessoryPart* the *PartRelationType* defines the semantic to resolve this reference for a resulting bill of material. If the type is *Mandatory* all referenced *PartVersions* shall be in the resulting bill of material. If the type is *Optional*, the referenced *PartVersions* can be selected by choice into the resulting bill of material. However, the choice applies to all *PartVersion*s of one *PartRelation.* For *Mandatory* it is semantically equivalent to have one *PartRelation* referencing N *PartVersions* or to have N *PartRelations*, each referencing one *PartVersion.* The *PartRelationType OneOfAll* defines, that exactly one of the referenced *PartVersions* shall be chosen for the resulting bill of material.

If the same *PartVersion* is referenced multiple times, each reference counts as its own position.

Example: To express that a *PartVersion* shall be used at least three times and with a maximum of 6 times, three mandatory and three optional *PartRelations* to this *PartVersion* would be created.

With these concepts, simple yes/no decisions can be represented. However, there cases where there are constraints between accessory parts (e.g. if part A, then choice of 2 x B or 1 x C). To express such logic in a static object model is not very feasible and inflexible. For such cases, the *relationType 'Custom'* was introduced. In this case, the relationships and constraints between all referenced *accessoryPart* can be expressed with some custom expression language in the *customRelationExpression* attribute. Even if it is custom, the expression shall only refer to elements that are contained in the *accessoryPart* relation and shall not influence other *PartRelations* of the same *GeneralTechnicalPartSpecification*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

relationType	PartRelationType	1	Specifies the type of the relation.
customRelationExpression	String	0..1	Defines the relationship between the accessory parts in a proprietary expression language. This attribute shall only be used, if the *relationType = 'Custom'.*

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartVersion	accessoryPart	1..*	0..*	N	References the PartVersions that are related by the PartRelation.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ExtensionSlot	0..*	allowedInserts	0..*	N	References the *PartRelations* that are valid inserts for this *ExtensionSlot.* This reference points to *PartRelations* in order to allow referencing indirectly a *PartVersion* if the description of individual *PartVersions* is done with one physical VEC file per *PartVersion* and to allow the expression of optional inserts, choices etc. However, inserts for an *ExtensionSlot* are always *EEComponents* by itself. Therefore, the referenced *PartVersion* shall have a *PrimaryPartType = EEComponent.*
GeneralTechnicalPartSpecification	1	partRelation	0..*	Y	Specifies possible relations (accessories) of the specified part with other PartVersion (e.g. caps, clips).
Slot	0..*	supplementaryParts	0..*	N	References the *PartRelations* that specify supplementary parts for this slot.
ModularSlot	0..*	allowedInserts	0..*	N	References the *PartRelations* that are valid inserts for this *ModularSlot.* This reference points to *PartRelations* in order to allow referencing indirectly a *PartVersion* if the description of individual *PartVersions* is done with one physical VEC file per *PartVersion* and to allow the expression of optional inserts, choices etc. However, inserts for a *ModularSlot* are always ConnectorHousings by itself. Therefore, the referenced *PartVersion* shall have a *PrimaryPartType = ConnectorHousing*

7.6.41 Class PinComponent

A PinComponent describes a pin of an EEComponent. A PinComponent is part of a HousingComponent and is defined by tree aspects.

- Its geometric position in the *HousingComponent*, which is defined by the *referencedCavity.*
- Its physical electrical properties, which are defined by the referenced *TerminalSpecification.*
- Its electrical behaviour, which is defined configuration dependent by its *PinComponentBehaviour.*

General Information

Base Classifier	ConfigurableElement ExtendableElement

Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Identification of the PinComponent, which must be distinct for all PinComponent of a HousingComponents.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the *PinComponent*.
pinComponentType	PinComponentType	0..1	Specifies the type of a *PinComponent*.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TerminalSpecification	pinSpecification	0..1	0..*	N	References the TerminalSpecification describing the electrical connectivity aspect of the PinComponent. (see KBLFRM-300)
Cavity	referencedCavity	0..1	0..*	N	Defines the cavity in the corresponding ConnectorHousingSpecification of the HousingComponent where the PinComponent is located. (see KBLFRM-300)
PinComponentBehavior	behaviour	0..*	1	Y	Specifies the configuration dependent electrical behavior of the PinComponent.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
DiodeSpecification	0..*	anode	0..1	N	
PinComponentReference	0..*	pinComponent	1	N	Points to the PinComponent referenced by the PinComponent reference. (KBLFRM-401)
HousingComponent	1	pinComponent	0..*	Y	Specifies the PinComponents of HousingComponent. (see KBLFRM-300)
FuseComponent	0..*	connectedPins	0..2	N	
InternalComponentConnection	0..*	pins	2..*	N	
DiodeSpecification	0..*	cathode	0..1	N	

7.6.42 Class PinComponentBehavior

A *PinComponentBehavior* specifies the electrical behavior of a *PinComponent.* Since the behavior of a pin is configuration dependent (e.g. the software deployed on an ECU) the *PinComponentBehavior* inherits from *ConfigurableElement.* Therefore, a *PinComponent* can specify multiple *PinComponentBehavior.*

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Identification of the PinComponentBehavior which identifies it from a user perspective, and which must be distinct for all PinComponentBehaviors of a PinComponent.
signalDirection	SignalDirection	0..1	The direction of the signal on this pin.
pinType	PinType	0..1	Specifies the *PinType* of the *PinComponent*.
applianceType	PinApplianceType	0..1	Classifies the appliance of a Pin in terms of the duration of the appliance (see PinApplianceType).
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the *PinComponentBehaviour*.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinCurrentInformation	currentInformation	0..*	1	Y	Specifies the current information of the *PinComponent* in this *PinComponentBehavior*. Since the current values of a pin can be defined for different types and times it is possible to define multiple *PinCurrentInformations* for a *PinComponentBehavior*.
PinVoltageInformation	voltageInformation	0..*	1	Y	Specifies the voltage information of the *PinComponent* in this *PinComponentBehavior*. Since the voltage values of a pin can be defined for different types and times it is possible to define multiple *PinVoltageInformations* for a *PinComponentBehavior*.
Signal	signal	0..1	0..*	N	Specifies the *Signal* associated with the pin in this behavior.
PinOpticalInformation	opticalInformation	0..*	1	Y	Specifies the optical information of the pin, if it has the type optical.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PinComponent	1	behaviour	0..*	Y	Specifies the configuration dependent electrical behavior of the *PinComponent.*

7.6.43 Class PinCurrentInformation

Allows the definition of currents for a pin of an EEComponent. A current can be further specified by a duration. Attributes of the type PinCurrentInformation normally have the multiplicity [0..*]. This means that such an attribute can have PinCurrentInformation entries for different types and durations. It must not have multiple entries for the same type and duration.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	PinCurrentType	1	Defines the type of the current.
current	NumericalValue	1	The current of the pin.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinTiming	timing	0..*	0..1	Y	Specifies the timing of the *PinCurrentInformation.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PinComponentBehavior	1	currentInformation	0..*	Y	Specifies the current information of the *PinComponent* in this *PinComponentBehavior*. Since the current values of a pin can be defined for different types and times it is possible to define multiple *PinCurrentInformations* for a *PinComponentBehavior*.

7.6.44 Class PinOpticalInformation

Allows the specification of optical information in a PinComponentBehavior.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
frequency	NumericalValue	1	The frequency of the optical signal.
attenuation	NumericalValue	0..1	The attenuation of the optical pin at the defined frequency.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PinComponentBehavior	1	opticalInformation	0..*	Y	Specifies the optical information of the pin, if it has the type optical.

7.6.45 Class PinTiming

Specifies the timing for a *PinCurrentInformation* or a *PinVoltageInformation*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	PinTimingType	1	Specifies the type of the timing.
time	NumericalValue	1	Specifies the time value of the timing.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PinCurrentInformation	0..1	timing	0..*	Y	Specifies the timing of the *PinCurrentInformation.*
PinVoltageInformation	0..1	timing	0..*	Y	Specifies the timing of the *PinVoltageInformation*.

7.6.46 Class PinVoltageInformation

Allows the definition of voltages for a pin of an *EEComponent*. A current can be further specified by a duration. Attributes of the type *PinVoltageInformation* normally have the multiplicity [0..*]. This means that such an attribute can have *PinVoltageInformation* entries for different types and durations. It must not have multiple entries for the same type and duration.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	PinVoltageType	1	Defines the type of the voltage.
voltage	NumericalValue	1	The voltage of the pin.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinTiming	timing	0..*	0..1	Y	Specifies the timing of the *PinVoltageInformation*.

Incoming Relations

Other End	This End	General

Type	Mult	Role	Mult	Agg	Comment
PinComponentBehavior	1	voltageInformation	0..*	Y	Specifies the voltage information of the *PinComponent* in this *PinComponentBehavior*. Since the voltage values of a pin can be defined for different types and times it is possible to define multiple *PinVoltageInformations* for a *PinComponentBehavior*.

7.6.47 Class PluggableTerminalSpecification

Specification for the definition of pluggable terminals.

General Information

Base Classifier	TerminalSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
terminalType	PluggableTerminalType	0..1	Defines the type of the terminal. The type defines constraints about the numbers of wire and terminal receptions and their relations.

7.6.48 Class PotentialDistributorSpecification

Specifies the properties of a potential distributor (e.g. a joint connector).

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
screwable	Boolean	0..1	Defines if the *PotentialDistributor* can be screwed by itself to be fixed. This fixation is a non-electrical connection. If the *PotentialDistributor* can be fixated electrical (like a ring terminal) it shall have one *PinComponent* with a *RingTerminalSpecification*.
boltDiameter	NumericalValue	0..1	Specifies the diameter of the bolt for which the potential distributor is designed for.
boltType	String	0..1	Specifies the type of the bolt.

7.6.49 Class PowerConsumption

Defines the power consumption of an *EEComponent*. An *EEComponent* can have multiple different *PowerConsumptions* e.g. standby, maximum. An *EEComponent* can have multiple *PowerConsumptions* but must not have more than one *PowerConsumptions* of the same *type*.

General Information

Base Classifier	

Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	String		The type of a *PowerConsumption*. This should be an enumeration of values defined in a conformance class.
value	NumericalValue		Specifies the power consumption for this type of *PowerConsumption*.

7.6.50 Class RelaySpecification

Specification of the electrological aspects of a relay.

General Information

Base Classifier	EEComponentSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
iMax	NumericalValue	0..1	Specifies the maximum current allowed for the relais.
relaisType	RelaisType	0..1	The type of the relay (switching behaviour). This is an OpenEnumeration, for values see *RelaisType.*
lowNoise	Boolean	0..1	Defines if the relais switch with low noise / silently or not.
applianceType	RelaisApplianceType	0..1	Specifies the appliance type of a relais.

7.6.51 Class RingTerminalSpecification

Specification for the definition of ring terminals. These are the counterparts to bolt terminals.

General Information

Base Classifier	TerminalSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
boltDiameter	NumericalValue	0..1	Specifies the diameter of the bolt for which the ring terminal is designed for in a numerical way.
boltNominalSize	String	0..1	Defines the size (diameter) of the bolt for which the ring terminal is designed for in a nominal way (e.g. "M8").
thickness	NumericalValue	0..1	Specifies the thickness of the contact surface of the ring terminal.

boltType	String	0..1	Specifies the type of the bolt.
outsideDimension	NumericalValue	0..1	Specifies the diameter of the circle around the center of the bolt which passes through the farthest outside point of the contact surface of the ring terminal. (see KBLFRM-311)
torsionProtection	Boolean	0..1	Specifies if the ring terminal is torsion protected or not. (see KBLFRM-311)

7.6.52 Class SealedCavitiesAssignment

A *SealedCavitiesAssignment* groups the cavities of ONE connector that are sealed by a MultiCavityPlug. If a MultiCavityPlug fits into more than one connector, than there are as many *SealedCavitiesAssignments*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Cavity	sealedCavities	1..*	0..*	N	Specifies the Cavities that are sealed.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
MultiCavityPlugSpecification	1	assignment	0..*	Y	References the *SealedCavitiesAssignments* that are valid for this *MultiCavityPlug*. One individual *SealedCavitiesAssignment* is used for each connector housing that matches witch this *MultiCavityPlug.*

7.6.53 Class SegmentConnectionPoint

Specifies a point where the connector can be attached to the topology (sometimes called bundle position point or insertion point).

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the SegmentConnectionPoint. The identification is guaranteed to be unique within the ConnectorHousingSpecification.

Outgoing Relations

Other End	This	General

Type	Role	Mult	Mult	Agg	Comment
CavityAddOn	cavityAddOns	0..*	1	Y	
Cavity	reachableCavities	0..*	0..*	N	Specifies the *Cavities* that are reachable with wires through this *SegmentConnectionPoint.*
ModularSlotAddOn	ModularSlotAddOns	0..*	1	Y	
PlacementPoint	placementPoint	0..1	0..*	N	Specifies the *PlacementPoint* that represents this *SegmentConnectionPoint* in a *PlaceableElementSpecification.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConnectorHousingSpecification	1	segmentConnectionPoint	0..*	Y	Specifies the *SegmentConnectionPoints* the connector housing.

7.6.54 Class ShieldSpecification

Specifies the properties of a shield.

General Information

Base Classifier	ConductorSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
opticalTissueDensity	NumericalValue	0..1	Defines the optical tissue density of the strands of a shield. This is defined as a value in percentage.
windingType	String	0..1	Defines the type of winding of the shield. E.g. if the shield is a sheet, it can be folded around the inner elements and along them like a cigarette paper or it can be winded around them like the taping of a harness. Both types result in different manufacturing and EMC properties.

7.6.55 Class Slot

A slot is a group of cavities in a connector housing with own properties. The design of a slot is described in a SlotSpecification.

General Information

Base Classifier	AbstractSlot
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
colorInformation	Color	0..*	Specifies the color of the slot.

sealingType	SlotSealingType	0..1	Specifies the type of the sealing of the slot, if sealed. The values are defined in an *OpenEnumeration*.
permittedTerminalSupplierCompanyNames	String	0..*	If this attribute is defined, it is only permitted to use terminals of one of the listed terminal suppliers. The used company name shall be same as the one used as *PartVersion.companyName* for part numbers of this supplier.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Cavity	cavity	1..*	1	Y	Specifies the Cavities forming the Slot.
PartRelation	supplementaryParts	0..*	0..*	N	References the *PartRelations* that specify supplementary parts for this slot.

7.6.56 Class SlotLayout

For regularly laid out slots the slot layout describes the positions of the cavities

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
gridX	NumericalValue	0..1	The horizontal distance between the center points of two cavities.
gridY	NumericalValue	0..1	The vertical distance between the center points of two cavities.
rowCount	Integer	0..1	The number of cavity rows of the slot.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SlotSpecification	1	slotLayout	0..1	Y	References the layout associated with this slot.

7.6.57 Class SlotSpecification

Specification for the definition of slots.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

emvProtectionRequired	Boolean	0..1	Specifies whether the slot needs EMV protection.
gender	SlotGender	0..1	Specifies the gender of the slot. Valid values are defined in an open enumeration.
layoutType	String	0..1	Specifies the slot's layout type.
secondaryLocking	Boolean	0..1	Specifies whether the slot supports secondary locking.
numberOfCavities	Integer	1	The possible number of cavities in the layout defined by the SlotSpecification. This includes all cavities in the layout. The actual Slot can define specific cavities in the layout as "not available".

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
SlotLayout	slotLayout	0..1	1	Y	References the layout associated with this slot.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
AbstractSlot	0..*	slotSpecification	0..1	N	References the SlotSpecification that is satisfied by the slot.

7.6.58 Class SpliceTerminalSpecification

Specification for the definition of splice terminals.

General Information

Base Classifier	TerminalSpecification
Applied Stereotype	
Is Abstract	false

7.6.59 Class SwitchingState

A *SwitchingState* defines a certain static state of an *EEComponent*. Under which conditions or when such a state applies, should be described in an external description model.

A *SwitchingState* references a collection of *InternalComponentConnections*, with the semantic that these connections exist (and only these connections) when the switching state is active.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Identification of the SwitchingState, which must be distinct for all SwitchingStates of an EEComponent.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
InternalComponentConnection	switchedConnections	0..*	0..*	N	Specifies the *InternalComponentConnections* that are switched by this *SwitchingState.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
EEComponentSpecification	1	states	0..*	Y	Specifies the available *SwitchingStates* of the EEComponent.

7.6.60 Class TerminalCurrentInformation

Allows the definition of valid current ranges for a terminal. A current range is always defined for a coreCrossSectionArea and a nominal voltage. Attributes of the type CurrentRangeInformation normally have the multiplicity [0..*]. This means that such an attribute can have CurrentRangeInformation entries for different coreCrossSectionAreas and nominalVoltages. It must not have multiple entries for the same coreCrossSectionAreas and nominalVoltages.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
currentRange	ValueRange	1	Specifies the valid current range for the terminal. It is defined as a value range, because there are terminals where minimum current is needed to avoid corrosion.
nominalVoltage	NumericalValue	1	Specifies the nominalVoltage for which the CurrentRangeInformation is valid.
coreCrossSectionArea	NumericalValue	1	Specifies the coreCrossSectionArea for which the CurrentRangeInformation is valid. The coreCrossSectionArea is a relevant information, because the thermal absorption of the core depends on the cross-section area and thus is an influence factor for the valid current range.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TerminalSpecification	1	currentInformation	0..*	Y	Specifies the *TerminalCurrentInformation* that is applicable for the terminal.

7.6.61 Class TerminalReception

A TerminalReception is the area of a terminal where the contacting with another terminal (e.g. between a connector housing and a control unit) takes place. Normally the terminal reception is placed in a cavity of a connector housing.

This class represents such an area on the terminal.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the TerminalReception. The identification is guaranteed to be unique within the TerminalSpecification (this might be for example a reception number).
gender	String	0..1	Specifies the gender of the TerminalReception. The gender is included in the TerminalReception class in order to be able to refer the same TerminalReceptionSpecification gender independent.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TerminalReceptionSpecification	terminalReceptionSpecification	0..1	0..*	N	References the TerminalReceptionSpecification that specifies the TerminalReception.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
InternalTerminalConnection	0..1	terminalReception	0..*	N	References the TerminalReceptions that participate in the InternalTerminalConnection.
TerminalSpecification	1	terminalReception	0..*	Y	Specifies the TerminalReceptions of the terminal described by the TerminalSpecification.
TerminalReceptionReference	0..*	terminalReception	1	N	References the *TerminalReception* that is instanced by this *TerminalReceptionReference.*

7.6.62 Class TerminalReceptionSpecification

Specification for the definition of terminal receptions. A TerminalReception is the area of a terminal where the contacting with another terminal (e.g. between a connector housing and a control unit) takes place. Normally the terminal reception is placed in a cavity of a connector housing.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
cavityDesign	String	0..1	Deprecated (since Version 1.1.4): This attribute has been marked as deprecated, as it has been replaced by the more meaningful mechanism with *TerminalTypes.*

platingMaterial	Material	0..*	Specifies the plating material of the terminal reception.
primaryLockingType	PrimaryLockingType	0..1	Specifies if the terminal reception has a primary locking and of what type it is.
pullOutForce	NumericalValue	0..1	The force until the terminal is pulled out of the housing (normally a not intended case). KBLFRM-366

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TerminalType	terminalType	0..1	0..1	Y	Specifies the terminal type that is associated with the terminal reception.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TerminalReception	0..*	terminalReception Specification	0..1	N	References the TerminalReceptionSpecification that specifies the TerminalReception.

7.6.63 Class TerminalSpecification

Specification for the definition of terminals. A terminal can own multiple WireReceptions & TerminalReceptions.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
voltageRange	ValueRange	0..1	Specifies the allowed voltage range for the connector housing.
sealingType	TerminalSealingType	0..1	Defines the *SealingType* of the Terminal. This type always refers to the sealing of the terminal itself. However, even a terminal which is not sealable can be used in sealed locations with additional measures (e.g. on the slot).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
InternalTerminalConnection	internalTerminalConnection	0..*	1	Y	Specifies the InternalTerminalConnections of the terminal.
WireReception	wireReception	0..*	1	Y	Specifies the WireReceptions of the terminal described by the TerminalSpecification.
TerminalReception	terminalReception	0..*	1	Y	Specifies the TerminalReceptions of the terminal described by the TerminalSpecification.
TerminalCurrentInformation	currentInformation	0..*	1	Y	Specifies the *TerminalCurrentInformation* that is applicable for the terminal.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PinComponent	0..*	pinSpecification	0..1	N	References the TerminalSpecification describing the electrical connectivity aspect of the PinComponent. (see KBLFRM-300)
TerminalRole	0..*	terminalSpecification	1	N	References the *TerminalSpecification* that is instanced by this *TerminalRole.*
Cavity		integratedTerminalSpecification	0..1	N	Specifies the terminal, if the cavity has an integrated terminal (e.g. an IDC).

7.6.64 Class TerminalType

Defines the type (system) of a terminal. The type is specified by a combination of a name for the system and an optional nominalSize.

In some processes the terminal type is referred as cavity system, because the system of cavities, terminals, seals, plugs and other cavity accessories must match / be compatible.

However, since a terminal has only one since type and a cavity can be compatible to more than one it is named *TerminalType* in the VEC.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
name	String	1	Specifies the type of the cavity (e.g. MQS, DFK, ...).
nominalSize	String	0..1	Specifies the nominal size of terminals that fit into the cavity. (e.g. 2x4).

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TerminalReceptionSpecification	0..1	terminalType	0..1	Y	Specifies the terminal type that is associated with the terminal reception.

7.6.65 Class WireElement

A *WireElement* specifies a *WireElementSpecification* in the context of a *WireSpecification*. This is necessary to define a unique *identification* of a *WireElementSpecification* in the context of a *WireSpecification.* Additionally, the *WireElement* serves as anchor for the instancing of a wire (*WireElementReference*) as the *WireElementSpecifications* are not uniquely related to a *WireSpecification* for reasons of reusability.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	

Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	The identification of the WireElement. The identification is guaranteed to be unique within a wire and immutable. The identification is guaranteed to be the same for the same WireElement over different VEC documents.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireElementSpecification	wireElementSpecification	1	0..*	N	Reference the *WireElementSpecification* that is represented by the *WireElement.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireElement	0..1	subWireElement	0..*	Y	Defines the *subWireElements* of this *WireElement*. The *subWireElements* shall be consistent with the *subWireElementSpecifications* of the *WireElementSpecification* referenced by this *WireElement* and shall resemble that hierarchy.
WireElementReference	0..*	referencedWireElement	1	N	References the WireElement that is represented by the WireElementReference.
WireSpecification	0..1	wireElement	1	Y	Specifies the *WireElement* that represents the root of the *WireSpecification*.

7.6.66 Class WireElementSpecification

A WireElementSpecification is the basic element to describe a wire in the VEC. A WireElementSpecification can be atomic or composed recursively out of other WireElementSpecifications. A WireElementSpecification can reference an InsulationSpecification, if it has an insulation, a CoreSpecification, if it has a core or a WireGroupSpecification if it is a grouping of other WireElementSpecifications in the Wire (e.g. a multi-core wire with twisted pairs).

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	WireType	0..*	Defines the type of the wire. A wire must not have more than one type. This attribute allows more than one value for the reason, that the same type can be expressed in multiple reference systems.
minBendRadiusDynamic	NumericalValue	0..1	Specifies the minimum bend radius for wire element, if it is used in a dynamic environment, where it is bended repeatedly (e.g. in grommet of the back door). (see KLBFRM-311)
minBendRadiusStatic	NumericalValue	0..1	Specifies the minimum bend radius for wire element, if it is used in a static environment, where it is bended once during installation.

			After that it remains unchanged in its bended position during usage. (see KLBFRM-311)
outsideDiameter	NumericalValue	0..1	Specifies the outside diameter of the WireElement.
suitedForDynamicUse	Boolean	0..1	Specifies if it is allowed to use the WireElement in a dynamic environment. (see KBLFRM-311)
impedance	NumericalValue	0..1	Defines the impedance of this wireElement. Typically used for WireElements that have subWireElements e.g. twisted pair or coax wires.
size	Size	0..1	Defines the size of a WireElement if it has not the shape of circle. If it has the shape of a circle the size is normally defined by its outside diameter.
validWireReceptionTypes	WireReceptionType	0..*	Defines the *WireReceptionTypes* that are allowed for joining with the specified *WireElement.*
gridSpacing	NumericalValue	0..1	Defines the grid spacing. The grid spacing is the distance between the centres of two adjacent sub wire elements. This attribute is only valid for *WireElementSpecifications* that have *SubWireElementSpecifications.*
shape	WireElementShape	0..1	Defines the shape of a *WireElement.* Circular wire elements are defined by their outsideDiameter, all others are defined by their size.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
FillerSpecification	fillerSpecification	0..1	0..*	N	If the *WireElement* is a filler then the specification of the filler is referenced here.
ConductorSpecification	conductorSpecification	0..1	0..*	N	If the *WireElement* has a core then the specification of the core is referenced here.
InsulationSpecification	insulationSpecification	0..1	0..*	N	If the *WireElement* has an insulation then the specification of the insulation is referenced here.
WireGroupSpecification	wireGroupSpecification	0..1	0..*	N	If the *WireElementSpecification* is representing a wire group, then the specification of the wire group is referenced here. That means as well, that the *WireElementSpecification* shall have *subWireElementSpecifications*.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireElement	0..*	wireElementSpecification	1	N	Reference the *WireElementSpecification* that is represented by the *WireElement.*
WireElementSpecification	0..*	subWireElementSpecification	0..*	N	Specifies the contained subWireElements if the WireElement has sub elements. If a *WireElementSpecification* contains the same *WireElementSpecification* multiple times, then it shall be referenced as often as it occurs in the reality. Otherwise the *WireElementSpecification* wouldn't specify a wire element unambiguously, because the representation in the model would be the same, regardless of the number of sub elements.

					Therefore, this association must not be realized with a "set" semantic.
WireSpecification	0..*	wireElementSpecification	1	N	References the *WireElementSpecification* that defines the properties of the top-most *WireElement.*

7.6.67 Class WireEndAccessorySpecification

WireEndAccessorySpecifications are describing parts that are used for a treatment of a wire end before the actual terminal is attached to the *WireEnd* (e.g. a ferrule).

The actual specification is done by the referenced *WireReceptionSpecification* since a *WireEndAccessory* is basically just a single *WireReception*.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireReceptionSpecification	wireReceptionSpecification	0..1		N	References the single *WireReceptionSpecification* that is specifying the properties of the *WireEndAccessory* that are related to wire.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireEndAccessory Role		wireEndAccessory Specification	1	N	References the *WireEndAccessorySpecification* that is instanced by this *WireEndAccessoryRole.*

7.6.68 Class WireGroupSpecification

Specification for the definition of WireGroups. Wire groups are for example used for the representation of twisted pairs.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
groupType	String	1	Specifies the type of the group (e.g. twisted pair, ...).
lengthOfTwist	NumericalValue	0..1	Specifies the length of twist if the wire group is representing a twisted pair.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireGrouping	0..*	wireGroupSpecification	0..1	N	References the *WireGroupSpecification* that applies to the *WireGrouping*.
WireElementSpecification	0..*	wireGroupSpecification	0..1	N	If the *WireElementSpecification* is representing a wire group, then the specification of the wire group is referenced here. That means as well, that the *WireElementSpecification* shall have *subWireElementSpecifications*.

7.6.69 Class WireReception

A WireReception is the area of a terminal where the contacting with a wire element (e.g. by crimping) takes place. This class represents such an area on the terminal. Its description is done with a WireReceptionSpecification, which is independent from the TerminalSpecification. This allows the reuse of the specification of similar WireReception on different terminals.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the WireReception. The identification is guaranteed to be unique within the TerminalSpecification (this might be for example a reception number).
angle	NumericalValue	0..2	Specifies the angles against two planes of the connector housing a wire reception can be buckled. The planes are defined in a way, that angles of [0.0, 0.0] specify an unbuckled (regular) terminal. The two planes are perpendicular to each other and parallel to the plugging direction of the terminal (the direction of the pin). The plane for the first angle is the plane in which the uncrimped wire reception would be, if the terminal has not been buckled. Since most terminals are cut or punched in some way out of a sheet of metal, this plane would be the same as metal sheet before further deformation. For ring terminals the first plane is perpendicular to the bolt direction. With a viewing direction from the wire reception to the terminal reception and the intended wire position above the terminal, buckling directions up and right are expressed by positive angles, down and left by negative angles.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireReceptionSpecification	wireReceptionSpecification	0..1	0..*	N	References the WireReceptionSpecification that specifies the WireReception.

PlacementPoint	placementPoint	0..1	0..*	N	Specifies the *PlacementPoint* that represents this *WireReception* in a PlaceableElementSpecification.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
InternalTerminalConnection	0..1	wireReception	0..*	N	References the WireReceptions that participate in the InternalTerminalConnection.
TerminalSpecification	1	wireReception	0..*	Y	Specifies the WireReceptions of the terminal described by the TerminalSpecification.
WireReceptionReference	0..*	wireReception	1	N	References the *WireReception* that is instanced by this *WireReceptionReference.*

7.6.70 Class WireReceptionAddOn

Specifies the wire addon required for this wire reception.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
wireAddOn	NumericalValue	1	Specifies the wire length add on needed for the wire reception.
type	WireAddOnType	1	Defines the type of the add-on.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireReceptionSpecification	1	addOns	0..*	Y	

7.6.71 Class WireReceptionSpecification

Specification for the definition of wire receptions. A WireReception is the area of a terminal where the contacting with a wire element (e.g. by crimping) takes place.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
coreCrossSectionArea	ValueRange	0..1	Specifies a value range for cross-section areas of cores that are compatible with the wire reception.

			For wire receptions where a single core is attached to, this value defines the *ValueRange* of the *crossSectionArea* of the core. For wire receptions where more than one core is attached to (e.g. ring terminals, splices), this attribute defines the valid value range for the sum of the cross-section areas of all attached cores.
insulationDisplacementLength	NumericalValue	0..1	Specifies the length of the insulation which must be stripped off to fit to this wire reception.
multiContact	Boolean	0..1	Specifies if it is possible to contact more than one core at the wire reception. If this attribute is set to true, the wire reception is allowed for more than one wire, but one wire is possible as well. If set to false, only one wire is allowed.
wireReceptionType	WireReceptionType	0..1	Specifies the type of the wire reception. The wireReceptionType defines how the wire is joined with the wire reception. This attribute is defined in an OpenEnumeration.
wireElementOutsideDiameter	ValueRange	0..1	Specifies a value range for valid diameters a WireElement must have to fit to the wire reception.
platingMaterial	Material	0..*	Specifies the plating material of the wire reception.
sealable	Boolean	0..1	Specifies if the wire reception can be sealed. (see KBLFRM-311)

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConductorMaterial	validConductorMaterials	0..*	1	Y	Specifies the materials of a conductor, that are valid to use with this *WireReceptionSpecification*. This material shall be matched against the *ConductorSpecification.material.*
WireReceptionAddOn	addOns	0..*	1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireEndAccessorySpecification		wireReceptionSpecification	0..1	N	References the single *WireReceptionSpecification* that is specifying the properties of the *WireEndAccessory* that are related to wire.
WireReception	0..*	wireReceptionSpecification	0..1	N	References the WireReceptionSpecification that specifies the WireReception.

7.6.72 Class WireSpecification

Specification for the definition of a wire. This can either be a complex multicore wire or a normal single core. In the VEC a wire is defined by one WireElement, which may be composed from other WireElements.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireElement	wireElement	1	0..1	Y	Specifies the *WireElement* that represents the root of the *WireSpecification*.
WireElementSpecification	wireElementSpecification	1	0..*	N	References the *WireElementSpecification* that defines the properties of the top-most *WireElement.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireRole	0..*	wireSpecification	1	N	References the *WireSpecification* that is instanced by this *WireRole.*

7.6.73 Class WireType

Specifies a wire type. A wire type is always defined by a key value. What wire type is meant by this key value is defined by a standard reference system.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	String	1	Specifies the type of the wire (e.g. FLRY, NYFAZw). Valid values are defined by the wireTypeReferenceSystem.
referenceSystem	String	1	Specifies the reference system for the wire type.

7.6.74 Enumeration CavityGeometry

Defines valid values of the geometry of cavities in the sealing area (crimp end).

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Square		
Circular		
Oval		

7.6.75 Enumeration ConnectorOutletDirection

Defines the *OutletDirection* of a connector for wires.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Straight		
Angled		

7.6.76 Enumeration FuseType

Defines the values for the type a fuse. This is the geometrical type.

General Information

Applied Stereotype	

Enumeration Literals

Name		Comment
Type_SF51		also known as: mega
Type_SF30		also known as: midi
Type_SF		also known as: strip
Type_F		also known as: mini
Type_C		also known as: ato
Type_E		also known as: maxi
Type_A1		also known as: jcase
Type_A1S		also known as: flat
Form_CB15_CatE		also known as: auto
MITOX		

7.6.77 Enumeration HousingComponentType

Defines the valid HousingComponentTypes. The values in this Enumeration are matching the relevant *PrimaryPartTypes.* For a description of the values see *PrimaryPartType.*

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
ConnectorHousing		

Fuse		
Relay		
EEComponent		
Terminal		
RingTerminal		
MultiFuse		

7.6.78 Enumeration PartRelationType

Defines how the set of *acessoryParts* referenced by a *PartRelation* should be interpreted.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Mandatory		*Mandatory* means that in a usage of a component all referenced *acessoryParts* must be used.
Optional		*Optional* means that in a usage of a component some referenced *acessoryParts* can be used by choice.
OneOfAll		*OneOfAll* means that in a usage of a component exactly one of the referenced *acessoryParts* must be selected.
Custom		Custom means, that there is some kind of custom constraints / logic between referenced *PartVersions* that has to be evaluated to determine the selected accessories.

7.6.79 Enumeration PinApplianceType

Classifies the appliance of a Pin in terms of the duration of the appliance.

E.g. the power supply pin of a power window has "shortTerm" PinApplianceType, in contrast to the head light which is a "long term".

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
shortTerm		
longTerm		

7.6.80 Enumeration PinComponentType

Specifies the type of a *PinComponent*.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Switch		
Coil		
Ground		
PowerSupply		*PowerSupply* defines a *PinComponent* that is used to supply the component itself with power (in contrast to *PowerDistribution*). *PinComponentBehaviours* of *PinComponents* with this type always have the *SignalDirection "In*".
PowerDistribution		*PowerDistribution* defines a *PinComponent* that is used to distribute power to other components (in contrast to *PowerSupply*). The semantic of this type depends on the *SignalDirection*. *In* means, that this *PinComponent* is used to supply power to the EEComponent for further distribution to other components. *Out* means that the *PinComponent* is a source of power for other EEComponents.
Signal		
NotConnected		

7.6.81 Enumeration PinCurrentType

Defines the different available current types of a pin.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
minCurrent		
maxCurrent		
typicalCurrent		
standbyCurrent		
initCurrent		
blockCurrent		
startStopCurrent		
overrunCurrent		
leakageCurrent		Defines the current that leaks at this pin.
deepSleepCurrent		Defines the current on the pin, when the EEComponent is in deep sleep mode.

clippingCurrent		Current at which the output driver of an ECU turns off or limits the current on an output in (a behavior similar to a fuse).

7.6.82 Enumeration PinTimingType

Defines the different available timing types of a pin.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
initTime		
startStopTime		
overrunTime		
blockTime		
currentPeakDistances		
duration		

7.6.83 Enumeration PinType

Defines the type of a pin.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
resistive		
inductive		
capacitive		
optical		
filament		

7.6.84 Enumeration PinVoltageType

Defines the different available voltage types of a pin.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
minVoltage		
maxVoltage		
typicalVoltage		
standbyVoltage		
initVoltage		
blockVoltage		
startStopVoltage		
overrunVoltage		

7.6.85 Enumeration PluggableTerminalType

Defines valid values for the type of PluggableTerminals. The type defines constraints about the numbers of wire and terminal receptions and their relations.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Standard		
Coax		
MultiContact		
Bridge		
PiggyBack		

7.6.86 Enumeration PrimaryLockingType

Defines the valid primary locking types for terminals.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Lance		

None		

7.6.87 Enumeration RelaisApplianceType

Specifies the appliance type of a relais.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
soldered		
screwed		
plugged		

7.6.88 Enumeration RelaisType

Defines the type of a relais (switching behaviour).

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
normally open		
normally closed		Defines a relais that is *closed* in its normal state.
switch		Defines a relais that is a switch.
bistable		Defines a relais that is *bistable*.

7.6.89 Enumeration SealingGeometry

Defines valid values of the geometry of a cavity sealing. It defines the cross-section geometry.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Circular		
Oval		
Square		

Segmented		
Sectored		

7.6.90 Enumeration SlotGender

Defines the gender of a slot.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Male		
Female		
Unspecified		

7.6.91 Enumeration SlotSealingType

Defines the possible sealing types for a slot, if the slot shall be sealed in its usage.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
None		The *Slot* cannot be sealed at all.
SingleSealing		The *Slot* is sealed with a single *Seal* per *Cavity* e.g. a *CavitySeal* or a *CavityPlug.*
MultiSealing		The *Slot* is sealed with a more complex sealing variant, typically sealing multiple cavities with a single seal (e.g. a *MultiCavityPlug* or a combination of those).
Moulded		The *Slot* is sealed by moulding it with some sort of sealing compound.

7.6.92 Enumeration TerminalSealingType

Defines the possible sealing types for a terminal, if the terminal shall be sealed in its usage.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
None		The Terminal cannot be sealed by itself (e.g. a CavitySeal)

Sealed		The Terminal can be sealed and can only be used together with a cavity seal. It is not allowed to be used unsealed.
Neutral		The terminal can be sealed (with a cavity seal), but a seal is not mandatory for this terminal.
Moulded		The terminal can be used in a moulded slot.

7.6.93 Enumeration WireAddOnType

Specifies possible values for the t*ype* of wire add-ons (e.g. *CavityAddOn*).

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Contract		The agreed add-on for any negotiations and calculations.
Production		The add-on length for the use in production environments.

7.6.94 Enumeration WireElementShape

Defines the shape of a wire element.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Circular		
Flat		
Elliptical		

7.6.95 Enumeration WireReceptionType

The *WireReceptionType* defines in an *OpenEnumeration* how a wire is joined with a wire reception.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Crimp		
Soldering		
PlasmaSoldering		

SpotResistentWelding		
FrictionWelding		
UltraSonicWelding		
UltraSonicCompactation		UltraSonicCompactation, typically used for aluminium wires.

7.6.96 Primitive CodingName

General Information

Applied Stereotype	OpenPatternRestriction

7.7 Module external_mapping

7.7.1 Class ExternalMapping

An *ExternalMapping* is used to relate an *ExtendableElement* in the VEC with an element located in an external data source. The element in the VEC is referenced by the *mappedElement*, the external element is identified by the attribute *externalReference.*

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
externalReference	String	1	Defines the unique key of the external element. How this key shall be interpreted is dependent from the format of the external data source.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ExtendableElement	mappedElement	1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ExternalMappingSpecification	1	mappings	0..*	Y	Specifies the mappings of individual element.

7.7.2 Class ExternalMappingSpecification

An *ExternalMappingSpecification* is used to define a mapping between an external data source (represented by the referenced *mappedDocument*) and the content of a VEC file.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ExternalMapping	mappings	0..*	1	Y	Specifies the mappings of individual element.
DocumentVersion	mappedDocument	1	0..*	N	Reference to the *DocumentVersion* that represents the external data source that connected to the VEC content by the *ExternalMappingSpecification*.

7.8 Module geo_2d

7.8.1 Class BuildingBlockPositioning2D

Defines the position of a BuildingBlock2D on a HarnessDrawing.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint2D	centerPoint	0..1	0..1	Y	Specifies the center point of the BuildingBlock in the coordinate system of the harness drawing.
BuildingBlockSpecification2D	referenced2DBuildingBlock	1	0..*	N	References the building block which is placed on the harness drawing.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
HarnessDrawingSpecification2D	1	buildingBlockPositionings	1..*	Y	Specifies the BuildingBlockPositioning2Ds that are forming the 2D harness drawing.

7.8.2 Class BuildingBlockSpecification2D

Specification for the description of a two-dimensional building block. A building block is a reusable section of a geometry.

General Information

Base Classifier	Specification
Applied Stereotype	

Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
OccurrenceOrUsageViewItem2D	placedElementViewItem	0..*	1	Y	Specifies the view items for OccurrenceOrUsages on a BuildingBlockSpecification2D.
CartesianPoint2D	cartesianPoint	0..*	0..1	Y	Specifies the CartesianPoint2Ds that are used in the BuildingBlockSpecification2D.
CartesianDimension	boundingBox	1	0..1	Y	Specifies the size of the area described by the BuildingBlockSpecification2D in Cartesian dimensions.
GeometrySegment2D	geometrySegment	0..*	1	Y	Specifies the GeometrySegment2Ds defined by the BuildingBlockSpecification2D.
Unit	baseUnit	1	0..*	N	
GeometryNode2D	geometryNode	0..*	1	Y	Specifies the GeometryNode2Ds defined by the BuildingBlockSpecification2D.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockPositioning2D	0..*	referenced2DBuildingBlock	1	N	References the building block which is placed on the harness drawing.

7.8.3 Class CartesianDimension

The CartesianDimension specifies the size of an object in 2D-space.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
height	Double	1	Specifies the height of the object. The unit of this value is the baseUnit of containing BuildingBlockSpecification2D.
width	Double	1	Specifies the width of the object. The unit of this value is the baseUnit of containing BuildingBlockSpecification2D.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockSpecification2D	0..1	boundingBox	1	Y	Specifies the size of the area described by the BuildingBlockSpecification2D in Cartesian dimensions.

7.8.4 Class CartesianPoint2D

A CartesianPoint2D is a point that is defined by its coordinates in a rectangular two-dimensional Cartesian coordinate system.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
x	Double	1	Specifies the value of x-coordinate of the Cartesian point. The unit of this value is the baseUnit of containing BuildingBlockSpecification2D.
y	Double	1	Specifies the value of y-coordinate of the Cartesian point. The unit of this value is the baseUnit of containing BuildingBlockSpecification2D.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockSpecification2D	0..1	cartesianPoint	0..*	Y	Specifies the CartesianPoint2Ds that are used in the BuildingBlockSpecification2D.
BuildingBlockPositioning2D	0..1	centerPoint	0..1	Y	Specifies the center point of the BuildingBlock in the coordinate system of the harness drawing.
GeometryNode2D	0..*	cartesianPoint	1	N	References the CartesianPoint2D where the GeometryNode2D is located.
Transformation2D	0..*	origin	1	N	References the CartesianPoint2D that is the origin of the Transformation2D.
PathSegment	0..*	controlPoint	0..*	N	The ordered list of control points through which the PathSegment goes.

7.8.5 Class CartesianVector2D

A Cartesian vector in the two-dimensional space.

General Information

Base Classifier	CartesianVector
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
x	Double	1	Specifies the x-coordinate in 2D space.
y	Double	1	Specifies the y-coordinate in 2D space.

7.8.6 Class GeometryNode2D

A GeometryNode2D is the geometric representation of a TopologyNode in 2D-space.

General Information

Base Classifier	GeometryNode
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint2D	cartesianPoint	1	0..*	N	References the CartesianPoint2D where the GeometryNode2D is located.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
GeometrySegment 2D	0..*	endNode	1	N	References the GeometryNode2D where the GeometrySegment2D ends.
GeometrySegment 2D	0..*	startNode	1	N	References the GeometryNode2D where the GeometrySegment2D starts.
BuildingBlockSpecification2D	1	geometryNode	0..*	Y	Specifies the GeometryNode2Ds defined by the BuildingBlockSpecification2D.

7.8.7 Class GeometrySegment2D

A GeometrySegment2D is the geometric representation of a TopologySegment in 2D-space.

General Information

Base Classifier	GeometrySegment
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
startVector	CartesianVector2D	1	Specifies the start vector of the geometry segment. The start vector is a tangent to the segment at the start position.
endVector	CartesianVector2D	1	Specifies the end vector of the geometry segment. The end vector is a tangent to the segment at the end position.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
GeometryNode2D	endNode	1	0..*	N	References the GeometryNode2D where the GeometrySegment2D ends.
GeometryNode2D	startNode	1	0..*	N	References the GeometryNode2D where the GeometrySegment2D starts.

PathSegment	pathSegment	0..*	1	Y	Specifies an ordered list of PathSegments that describe the appearance of the GeometrySegment2D. The appearance is described by the concatenation of the PathSegments beginning at the startNode of the GeometrySegment2D.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockSpecification2D	1	geometrySegment	0..*	Y	Specifies the GeometrySegment2Ds defined by the BuildingBlockSpecification2D.

7.8.8 Class HarnessDrawingSpecification2D

The HarnessDrawingSpecification2D specifies a two-dimensional drawing of a harness. A harness drawing is composed of one or more BuildingBlockSpecifaction2D which are placed on the drawing.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
BuildingBlockPositioning2D	buildingBlockPositionings	1..*	1	Y	Specifies the BuildingBlockPositioning2Ds that are forming the 2D harness drawing.

7.8.9 Class OccurrenceOrUsageViewItem2D

An OccurrenceOrUsageViewItem2D specifies the representation of an OccurrenceOrUsage in a 2DDrawing.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the OccurrenceOrUsageViewItem2D. The identification is guaranteed to be unique within the BuildingBlockSpecification2D.
aliasId	AliasIdentification	0..*	Specifies additional identifiers for the OccurrenceOrUsageViewItem2D.
gridSquare	String	0..1	Specifies the grid square in which the OccurrenceOrUsageViewItem2D is placed (e.g. E/40).

Outgoing Relations

Other End	This	General

Type	Role	Mult	Mult	Agg	Comment
Transformation2D	orientation	0..1	1	Y	Specifies the orientation of the view item.
OccurrenceOrUsage	occurrenceOrUsage	0..*	0..*	N	Specifies the *OccurrenceOrUsages* which are represented by the view item. **Important:** To use one *OccurenceOrUsageViewItem* for multiple *OccurrenceOrUsages* is only valid, if the referenced items are true alternatives to each other. That means, they must have an identical placement, the geometrical models used for each item must be substitutable and the item must be mutually exclusive to each other.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockSpecification2D	1	placedElementViewItem	0..*	Y	Specifies the view items for OccurrenceOrUsages on a BuildingBlockSpecification2D.

7.8.10 Class PathSegment

A PathSegment is a part of the 2D presentation of a GeometrySegment2D. The complete presentation of a GeometrySegment2D is built from an ordered list of PathSegments. Each PathSegment has an ordered list of control points through which the path goes. If no curveRadius is specified the control points are connected by a direct straight line. If a curveRadius is specified, the PathSegment can be drawn by a segment of a circle which touches all control points and has the radius specified.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
curveRadius	NumericalValue	0..1	The radius of the curve which describes the appearance of the path segment.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint2D	controlPoint	0..*	0..*	N	The ordered list of control points through which the PathSegment goes.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
GeometrySegment2D	1	pathSegment	0..*	Y	Specifies an ordered list of PathSegments that describe the appearance of the GeometrySegment2D. The appearance is described by the concatenation of the PathSegments beginning at the startNode of the GeometrySegment2D.

7.8.11 Class Transformation2D

A Transformation is a geometric transformation and specifies a transformation matrix.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
a11	Double	1	
a12	Double	1	
a21	Double	1	
a22	Double	1	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint2D	origin	1	0..*	N	References the CartesianPoint2D that is the origin of the Transformation2D.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
OccurrenceOrUsageViewItem2D	1	orientation	0..1	Y	Specifies the orientation of the view item.

7.9 Module geo_3d

7.9.1 Class BuildingBlockPositioning3D

Defines the position of a BuildingBlock3D in the HarnessGeometry.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Transformation3D	positioning	0..1	0..1	Y	Specifies the positioning of the building block in the harness geometry.

BuildingBlockSpecification3D	referenced3DBuildingBlock	1	0..*	N	References the building block that is positioned.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
HarnessGeometrySpecification3D	1	buildingBlockPositionings	0..*	Y	Specifies the BuildingBlockPositioning3Ds that are forming the HarnessGeometrySpecification3D.

7.9.2 Class BuildingBlockSpecification3D

Specification for the description of a three-dimensional building block. A building block is a reusable section of a geometry.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
GeometryNode3D	geometryNode	0..*	1	Y	Specifies the GeometryNode3Ds defined by the BuildingBlockSpecification3D.
CartesianPoint3D	cartesianPoint	0..*	1	Y	Specifies the CartesianPoint3Ds that are used in the BuildingBlockSpecification3D.
OccurrenceOrUsageViewItem3D	placedElementViewItem3D	0..*	1	Y	Specifies the view items for OccurrenceOrUsages in a BuildingBlockSpecification3D.
Unit	baseUnit	1	0..*	N	
TopologyZone		0..1	0..*	N	References the Zone that is building block represents. This shall be a TopologyZone with the type "DmuZone".
GeometrySegment3D	geometrySegment	0..*	1	Y	Specifies the GeometrySegment3Ds defined by the BuildingBlockSpecification3D.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockPositioning3D	0..*	referenced3DBuildingBlock	1	N	References the building block that is positioned.

7.9.3 Class CartesianPoint3D

A CartesianPoint3D is a point that is defined by its coordinates in a rectangular three-dimensional Cartesian coordinate system.

General Information

Base Classifier	ExtendableElement

Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
x	Double	1	Specifies the value of x-coordinate of the Cartesian point. The unit of this value is the baseUnit of containing BuildingBlockSpecification3D.
y	Double	1	Specifies the value of y-coordinate of the Cartesian point. The unit of this value is the baseUnit of containing BuildingBlockSpecification3D.
z	Double	1	Specifies the value of z-coordinate of the Cartesian point. The unit of this value is the baseUnit of containing BuildingBlockSpecification3D.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
LocalPosition	0..*	cartesianPoint	1	N	
Transformation3D	0..*	origin	1	N	Specifies the coordinates of the translation.
BuildingBlockSpecification3D	1	cartesianPoint	0..*	Y	Specifies the CartesianPoint3Ds that are used in the BuildingBlockSpecification3D.
GeometryNode3D	0..*	cartesianPoint	1	N	References the CartesianPoint3D where the GeometryNode3D is located.
LocalGeometrySpecification	0..1	cartesianPoint	0..*	Y	All *CartesianPoint3Ds* that are used in this *LocalGeometrySpecification.* All *CartesianPoint3Ds* are defined in relation to the coordinate system of the component.
NURBSControlPoint	0..*		1	N	The *CartesionPoint3D* that defines the position of this *NURBSControlPoint.*

7.9.4 Class CartesianVector3D

A Cartesian vector in three-dimensional space.

General Information

Base Classifier	CartesianVector
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
x	Double	1	Specifies the x-coordinate in 3D space.
y	Double	1	Specifies the y-coordinate in 3D space.
z	Double	1	Specifies the z-coordinate in 3D space.

7.9.5 Class Curve3D

The *Curve3D* is an abstract representation of a curve, that defines the three-dimensional appearance of the centreline of a segment. The concrete type of the curve (e.g. NURBSCurve) defines the mathematical function that applies to curve and stores the required parameter set for this function in the VEC.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
GeometrySegment 3D		curve	0..*	Y	Specifies an ordered list of curves which describe the centerline of the segment. If a segment is described by more than one curve, the centrelines of the individual curves are aligned in the order of this association.

7.9.6 Class GeometryNode3D

A GeometryNode3D is the geometric representation of a TopologyNode in 3D-space.

General Information

Base Classifier	GeometryNode
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint3D	cartesianPoint	1	0..*	N	References the CartesianPoint3D where the GeometryNode3D is located.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockSpecification3D	1	geometryNode	0..*	Y	Specifies the GeometryNode3Ds defined by the BuildingBlockSpecification3D.
GeometrySegment 3D	0..*	endNode	1	N	References the GeometryNode3D where the GeometrySegment3D ends.
GeometrySegment 3D	0..*	startNode	1	N	References the GeometryNode3D where the GeometrySegment3D starts.

7.9.7 Class GeometrySegment3D

A GeometrySegment3D is the geometric representation of a TopologySegment in 3D-space.

General Information

Base Classifier	GeometrySegment
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
startVector	CartesianVector3D	1	Specifies the start vector of the geometry segment. The start vector is a tangent to the segment at the start position.
endVector	CartesianVector3D	1	Specifies the end vector of the geometry segment. The end vector is a tangent to the segment at the end position.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
GeometryNode3D	endNode	1	0..*	N	References the GeometryNode3D where the GeometrySegment3D ends.
Curve3D	curve	0..*		Y	Specifies an ordered list of curves which describe the centerline of the segment. If a segment is described by more than one curve, the centrelines of the individual curves are aligned in the order of this association.
GeometryNode3D	startNode	1	0..*	N	References the GeometryNode3D where the GeometrySegment3D starts.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockSpecification3D	1	geometrySegment	0..*	Y	Specifies the GeometrySegment3Ds defined by the BuildingBlockSpecification3D.

7.9.8 Class HarnessGeometrySpecification3D

The HarnessGeometrieSpecification3D specifies a three-dimensional model of a harness. A harness model is composed of one or more BuildingBlockSpecifaction3D which are placed in the model.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	GeometryType	1	Specifies the type of the harness geometry.

Outgoing Relations

Other End	This	General

Type	Role	Mult	Mult	Agg	Comment
BuildingBlockPositioning3D	buildingBlockPositionings	0..*	1	Y	Specifies the BuildingBlockPositioning3Ds that are forming the HarnessGeometrySpecification3D.

7.9.9 Class NURBSControlPoint

Represents a control point of a *NURBSCurve.* It consists of a referenced *CartesianPoint3D* for the position and a *weight*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
weight	Double	1	The weight of the NURBSControlPoint.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint3D		1	0..*	N	The *CartesionPoint3D* that defines the position of this *NURBSControlPoint.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
NURBSCurve		controlPoint	0..*	Y	Defines the control points of the NURBS curve (for details see the class description of *NURBSCurve)*

7.9.10 Class NURBSCurve

The NURBSCurve represents the parameter set of a NURBS (Non-Uniform rational B-Spline) in the VEC. For a complete definition of NURBS see for example (https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline).

Basically, a NURBS curve is defined by:

- a **degree**
- a list of **control points**: with at least *degree + 1* points.
- a **weight** for each control point.
- a **knots** vector: a list of numbers, with (*degree + #controlpoints +1)* elements. Every number must be equal or greater than its predecessor, and the same value must not be repeated more than *degree* times. It seems that modern NURBS algorithms just require (*degree + #controlpoints -1)* control points.

Commonly used default assignments for the parameters are:

- **weight = 1** for all control points: In this case the curve is called "non-rational".
- **knot** vector: equidistant and increasing values in the knot vector (e.g. 1,2,3,4,5,6,7) means the curve is "uniform" which exists in two variants.

- **clamped** (or pinned): If the knot vector starts and ends with *degree* times the same value (e.g. degree = 3, knots = [0,0,0,1,2,3,4,5,5,5], then it is *clamped.* This has the effect, that the first and last control point coincide with the start and end point of the curve.
- **unclamped** (or unpinned): If there are no repeated values in the knot vector (e.g. 1,2,3,4,5,6,7) it is unclamped.

The VEC *NURBSCurve* corresponds to removed *BSplineCurve* (VEC Version <= 1.1.3 and KBL). However, the *BSplineCurve* did not define weight and knot vector, so default assignments where assumed. Existing implementations are using "uniform non-rational b splines", unfortunately some implementations use "uniform clamped" and some "unclamped".

Since a NURBS cannot be rendered correctly without the knowledge of all parameters and to avoid further misconceptions the VEC allows the definition of all parameters of a NURBS. Furthermore, it requires the specification of all parameters, even if some known default assignment (e.g. non-rational) is used.

General Information

Base Classifier	Curve3D
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
degree	Integer	1	Defines the degree of the NURBS (for details see the class description of NURBSCurve).
knots	Double	0..*	Defines the knot-vector of the NURBS(for details see the class description of *NURBSCurve).*

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
NURBSControlPoint	controlPoint	0..*		Y	Defines the control points of the NURBS curve (for details see the class description of *NURBSCurve)*

7.9.11 Class OccurrenceOrUsageViewItem3D

An *OccurrenceOrUsageViewItem3D* specifies the existence and representation of an *OccurrenceOrUsage* in a 3D-model.

The definition of the existence is necessary because a 150% model of a harness might contain different geometric variants and not all of them must be represented in the same *BuildingBlockSpecification3D*.

There are two different cases for the representation of *OccurrenceOrUsage*s in a 3D model. There are components that can be represented explicitly by a 3D model of the component (e.g. connectors, cable ducts, fixings) and there are components that are represented implicitly by a generic visualization and their placement on the topology (e.g. tapes, tubes). If an OccurrenceOrUsage has a 3D model and it shall be placed explicitly in 3D space, the *OccurrenceOrUsageViewItem3D* defines a *Transformation3D* in the *orientation* role. If no *orientation* is defined the *OccurrenceOrUsage* is represented implicitly.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the OccurrenceOrUsageViewItem. The identification is guaranteed to be unique within the BuildingBlockSpecification3D.
aliasId	AliasIdentification	0..*	Specifies additional identifiers for the OccurrenceOrUsageViewItem3D.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
OccurrenceOrUsage	occurrenceOrUsage	0..*	0..*	N	Specifies the *OccurrenceOrUsages* which are represented by the view item. **Important:** To use one *OccurenceOrUsageViewItem* for multiple *OccurrenceOrUsages* is only valid, if the referenced items are true alternatives to each other. That means, they must have an identical placement, the geometrical models used for each item must be substitutable and the item must be mutually exclusive to each other.
Transformation3D	orientation	0..1	0..1	Y	Specifies the orientation of the view item.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BuildingBlockSpecification3D	1	placedElementViewItem3D	0..*	Y	Specifies the view items for OccurrenceOrUsages in a BuildingBlockSpecification3D.

7.9.12 Class Transformation3D

A Transformation is a geometric transformation and specifies a transformation matrix.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
a11	Double	1	
a12	Double	1	
a13	Double	1	
a21	Double	1	
a22	Double	1	
a23	Double	1	

a31	Double	1	
a32	Double	1	
a33	Double	1	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint3D	origin	1	0..*	N	Specifies the coordinates of the translation.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
LocalGeometrySpecification	0..1	boundingBoxPositioning	0..1	Y	The transformation that defines the positioning of the bounding box in coordinate system of the component.
BuildingBlockPositioning3D	0..1	positioning	0..1	Y	Specifies the positioning of the building block in the harness geometry.
OccurrenceOrUsageViewItem3D	0..1	orientation	0..1	Y	Specifies the orientation of the view item.

7.9.13 Enumeration GeometryType

Enumeration for the definition of the GeometryType.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Dmu		The Geometry is a DMU model of the vehicle.
Formboard		The Geometry is a model of a Formboard.

7.10 Module instancing

7.10.1 Class OccurrenceOrUsage

An OccurrenceOrUsage is an abstract appearance of a part in the harness. This can either be a concrete part (with a part number or something similar) or the description (specification / requirements) of a part that should be used at that position. In the first case it would be a PartOccurrence in the second case a PartUsage.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the OccurrenceOrUsage. The identification is guaranteed to be unique within the context. For all VEC-documents an OccurrenceOrUsage-instance can be trusted to be the same if the context-instance is the same and the identification of the OccurrenceOrUsage is the same.
aliasId	AliasIdentification	0..*	Room to specify additional identifiers for the OccurrenceOrUsage.
abbreviation	LocalizedString	0..*	Specifies an abbreviation of the *OccurrenceOrUsage*. Normally this a human readable short name.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the OccurrenceOrUsage.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Role	role	0..*	0..1	Y	Specifies the different roles of the OccurrenceOrUsage.
UsageNode	realizedUsageNode	0..1	0..*	N	References the *UsageNode* that is realized by this *OccurrenceOrUsage*.
Instruction	installationInstruction	0..*	0..1	Y	Room to specify InstallationInstruction(s) for the OccurrenceOrUsage.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PartStructureSpecification	0..*	inBillOfMaterial	0..*	N	References the PartOccurrences that are building the bill of material of a composite part.
OccurrenceOrUsageViewItem2D	0..*	occurrenceOrUsage	0..*	N	Specifies the *OccurrenceOrUsages* which are represented by the view item. **Important:** To use one *OccurenceOrUsageViewItem* for multiple *OccurrenceOrUsages* is only valid, if the referenced items are true alternatives to each other. That means, they must have an identical placement, the geometrical models used for each item must be substitutable and the item must be mutually exclusive to each other.
OccurrenceOrUsageViewItem3D	0..*	occurrenceOrUsage	0..*	N	Specifies the *OccurrenceOrUsages* which are represented by the view item. **Important:** To use one *OccurenceOrUsageViewItem* for multiple *OccurrenceOrUsages* is only valid, if the referenced items are true alternatives to each other. That means, they must have an identical placement, the geometrical models used for each item must be substitutable and the item must be mutually exclusive to each other.
PartWithSubComponentsRole	0..*	subComponent	0..*	N	References the subcomponents that belong to this instance of a PartWithSubComponents.
OccurrenceOrUsage	0..*	referenceElement	0..*	N	References the *OccurrenceOrUsage* for which this *OccurrenceOrUsage* is an accessory / supplementary component.

ModuleList	0..*	completionComponents	1..*	N	References the components that are used as completion, if any of the Modules in the ModuleList appears in a configuration.

7.10.2 Class PartWithSubComponentsRole

A PartWithSubComponentsRole defines the instance specific properties and relationships of a part with subcomponents. A PartWithSubComponents is a composite part like an Assembly, a Module, Harness.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
OccurrenceOrUsage	subComponent	0..*	0..*	N	References the subcomponents that belong to this instance of a PartWithSubComponents.
PartStructureSpecification	partStructureSpecification	1	0..*	N	References the *PartStructureSpecification* that is instantiated by this *PartWithSubComponentsRole*.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ModuleList	0..*	moduleInList	1..*	N	References the Modules that belong to the ModuleList. If any of the referenced Modules participates in a configuration the completion components participate, too.
ModuleFamily	0..*	moduleInFamily	1..*	N	References the Modules that belong to the ModuleFamily.

7.10.3 Class Role

A Role is the corresponding mechanism for OccurrenceOrUsages to the PartOrUsageRelatedSpecifcations for PartVersions or PartUsages. The PartOrUsageRelatedSpecifcations are describing a certain aspect of the master data of a part. A Role describes the corresponding properties and relationships for instances of a part (e.g. the usage specific properties of a wire occurrence like the length or the contacting).

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	**Type**	**Mult**	**Comment**
identification	String	0..1	Specifies a unique identification of the Role. The identification is guaranteed to be unique within the OccurrenceOrUsage.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
OccurrenceOrUsage	0..1	role	0..*	Y	Specifies the different roles of the OccurrenceOrUsage.

7.10.4 Class SpecificRole

A SpecificRole is the possibility to define instance specific properties with custom properties (see ExtendableElement). This is necessary, if the part is described by custom properties of a PartOrUsageRelatedSpecification.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
specificRoleType	String	1	Specifies the type for role.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartOrUsageRelatedSpecification	specification	1	0..*	N	References the *PartOrUsageRelatedSpecification* that is instantiated by this *SpecificRole.*

7.11 Module instancing_electrical_parts

7.11.1 Class AbstractSlotReference

An *AbstractSlotReference* represents the usage of an AbstractSlot in the context of PartUsage or PartOccurrence.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the SlotReference. The identification is guaranteed to be unique within the ConnectorHousingRole. The cavity & slot number is defined by the associated cavity and slot.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

AbstractSlot	referencedSlot	1	0..*	N	Points to the slot referenced by the slot reference.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SlotCoupling	0..*	secondSlot	1	N	
ConnectorHousingRole	1	slotReference	0..*	Y	Specifies the SlotReferences used in the ConnectorHousingRole.
SlotCoupling	0..*	firstSlot	1	N	

7.11.2 Class BoltTerminalRole

A *BoltTerminalRole* defines the instance specific properties and relationships of a ring terminal.

General Information

Base Classifier	TerminalRole
Applied Stereotype	
Is Abstract	false

7.11.3 Class BridgeTerminalRole

A *BridgeTerminalRole* defines the instance specific properties and relationships of a bridge terminal (see BridgeTerminalSpecification).

A bridge terminal is a part that behaves like terminal but has no *WireReceptions.* It is used to create short circuit between different pins in a connector. In its use, it can realize a schematic connection on its own and independently of other components.

General Information

Base Classifier	TerminalRole
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Connection	connection	0..1	0..*	N	References the Connection that is realized by this BridgeTerminalRole.

7.11.4 Class CavityAccessoryRole

A *CavityAccessoryRole* defines the instance specific properties and relationships of a cavity accessory.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavityAccessorySpecification	cavityAccessorySpecification	1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CavityMounting	0..*	cavityAccessory	0..*	N	

7.11.5 Class CavityPlugRole

A CavityPlugRole defines the instance specific properties and relationships of a cavity plug.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavityReference	pluggedCavityRef	0..*	0..*	N	Defines which cavity / cavities in a connector instance is sealed by the plug.
CavityPlugSpecification	cavityPlugSpecification	1	0..*	N	References the *CavityPlugSpecification* that is instanced by this *CavityPlugRole.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CableLeadThroughReference	0..*	usedPlugs	0..*	N	References the plugs that are used with this CableLeadThroughReference. This association might be a 150% selection.
CavityMounting	0..*	replacedPlug	0..*	N	References the cavity plugs that are obsolete if the cavity mounting is realized.

7.11.6 Class CavityReference

A CavityReference represents the usage of a Cavity in the context of PartUsage or PartOccurrence.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

identification	String	0..1	Specifies a unique identification of the CavityReference. The identification is guaranteed to be unique within the ConnectorHousingRole. The cavity & slot number is defined by the associated cavity and slot.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TerminalRole	integratedTerminalRole	0..1	0..1	Y	Contains the terminal role if the cavity has an integrated terminal (e.g. an IDC).
ComponentPort	componentPort	0..1	0..*	N	References the *ComponentPort* that is implemented by this *CavityReference*.
Cavity	referencedCavity	1	0..*	N	Points to the cavity referenced by the cavity reference.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CavityMountingDetail	0..*	equippedCavityRef	1	N	References the cavity that is used for the detailed description of the cavity mounting.
CavityMounting	0..*	equippedCavityRef	1..*	N	References the cavities that are used for the cavity mounting.
CavityPlugRole	0..*	pluggedCavityRef	0..*	N	Defines which cavity / cavities in a connector instance is sealed by the plug.
SlotReference	1	cavityReference	0..*	Y	Specifies the CavityReferences used in the SlotReference.
CavityCoupling	0..*	secondCavity	1	N	
CavityCoupling	0..*	firstCavity	1	N	

7.11.7 Class CavitySealRole

A CavitySealRole defines the instance specific properties and relationships of a cavity seal.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavitySealSpecification	cavitySealSpecification	1	0..*	N	References the *CavitySealSpecification* that is instanced by this *CavitySealRole.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

CableLeadThrough Reference	0..*	usedSeals	0..*	N	
WireMounting	0..*	mountedCavitySeal	0..1	N	References the cavity seal that is used for the wire mounting.

7.11.8 Class ConnectorHousingCapRole

A *ConnectorHousingCapRole* defines the instance-specific properties of a cap for a connector (backshells).

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConnectorHousingCapSpecification	connectorHousingCapSpecification	1	0..*	N	References the *ConnectorHousingCapSpecification* that is instanced by this *ConnectorHousingCapRole.*

7.11.9 Class ConnectorHousingRole

A ConnectorHousingRole defines the instance specific properties and relationships of a connector housing.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
sealState	SealState	0..1	Specifies if this instance of a connector housing should be sealed (waterproof).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
AbstractSlotReference	slotReference	0..*	1	Y	Specifies the SlotReferences used in the ConnectorHousingRole.
ComponentNode	componentNode	0..1	0..*	N	References the ComponentNode that is realized by the referenced ConnectorHousing (OccurrenceOrUsage with ConnectorHousingRole). This can especially be relevant for inliners. KBLFRM-341.
ConnectorHousingSpecification	connectorHousingSpecification	1	0..*	N	References the *ConnectorHousingSpecification* that is instanced by this *ConnectorHousingRole.*

Incoming Relations

Other End	This End	General

Type	Mult	Role	Mult	Agg	Comment
CouplingPoint	0..*	secondConnector	0..1	N	
ModularSlotReference	0..*	usedInserts	0..*	N	References the inserts that are used in this *ModularSlotReference.* More than one insert is valid in the case variant dependent equipment of the slot.
CouplingPoint	0..*	firstConnector	0..1	N	
HousingComponentReference	0..1	connectorHousingRole	0..1	Y	

7.11.10 Class EEComponentRole

An EEComponentRole defines the instance specific properties and relationships of an EE-component.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ComponentNode	componentNode	0..1	0..*	N	References the ComponentNode that is realized by the referenced EEComponent (OccurrenceOrUsage with EEComponentRole). KBLFRM-341
EEComponentSpecification	EEComponentSpecification	1	0..*	N	References the *EEComponentSpecification* that is instanced by this *EEComponentRole.*
HousingComponentReference	housingComponentRef	0..*	1	Y	Specifies the HousingComponentReferences used in the EEComponentRole. (KBLFRM-401)
ExtensionSlotReference	extensionSlotRef	0..*	1	Y	Specifies the ExtensionSlotReferences used in the EEComponentRole.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ExtensionSlotReference	0..*	usedInserts	0..*	N	Defines the inserts used for extension slot in a defined instance. These can be more than one EEComponentRole, because of variance in a 150% specification.

7.11.11 Class ExtensionSlotReference

An ExtensionSlotReference represents the usage of an ExtensionSlot in the context of a PartUsage or PartOccurrence.

General Information

Base Classifier	ConfigurableElement

Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
EEComponentRole	usedInserts	0..*	0..*	N	Defines the inserts used for extension slot in a defined instance. These can be more than one EEComponentRole, because of variance in a 150% specification.
ExtensionSlot	extensionSlot	1		N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
EEComponentRole	1	extensionSlotRef	0..*	Y	Specifies the ExtensionSlotReferences used in the EEComponentRole.

7.11.12 Class HousingComponentReference

A HousingComponentReference represents the usage of a HousingComponent in the context of a PartUsage or PartOccurrence. (KBLFRM-401)

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the HousingComponentReference. The identification is guaranteed to be unique within the EEComponentRole.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinComponentReference	pinComponentRef	0..*	1	Y	Specifies the PinComponentReferences used in the HousingComponentReference. (KBLFRM-401)
ComponentConnector	componentConnector	0..1	0..*	N	References the ComponentConnector that is realized by the referenced HousingComponentReference.

HousingComponent	housingComponent	1	0..*	N	Points to the HousingComponent referenced by the HousingComponent reference. (KBLFRM-401)
ConnectorHousingRole	connectorHousingRole	0..1	0..1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
EEComponentRole	1	housingComponentRef	0..*	Y	Specifies the HousingComponentReferences used in the EEComponentRole. (KBLFRM-401)

7.11.13 Class ModularSlotReference

A *ModularSlotReference* represents the usage of a *ModularSlot* in the context of PartUsage or PartOccurrence.

General Information

Base Classifier	AbstractSlotReference
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConnectorHousingRole	usedInserts	0..*	0..*	N	References the inserts that are used in this *ModularSlotReference.* More than one insert is valid in the case variant dependent equipment of the slot.

7.11.14 Class OpenWireEndTerminalRole

An OpenWireEndTerminalRole defines the instance specific properties and relationships of an open wire end.

General Information

Base Classifier	TerminalRole
Applied Stereotype	
Is Abstract	false

7.11.15 Class PinComponentReference

A PinComponentReference represents the usage of a PinComponent in the context of a PartUsage or PartOccurrence. (KBLFRM-401)

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the PinComponentReference. The identification is guaranteed to be unique within the HousingComponentReference.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinComponent	pinComponent	1	0..*	N	Points to the PinComponent referenced by the PinComponent reference. (KBLFRM-401)
TerminalRole	terminalRole	0..1	0..1	Y	References the TerminalRole of PinComponentReference. This is required to specify a Mating for EEComponents with other EEComponents or a Harness. (KBLFRM-401)

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
HousingComponentReference	1	pinComponentRef	0..*	Y	Specifies the PinComponentReferences used in the HousingComponentReference. (KBLFRM-401)
PinWireMappingPoint		pinComponentReference	1	N	

7.11.16 Class PluggableTerminalRole

A PluggableTerminalRole defines the instance specific properties and relationships of a pluggable terminal.

General Information

Base Classifier	TerminalRole
Applied Stereotype	
Is Abstract	false

7.11.17 Class RingTerminalRole

A RingTerminalRole defines the instance specific properties and relationships of a ring terminal.

General Information

Base Classifier	TerminalRole
Applied Stereotype	
Is Abstract	false

7.11.18 Class SlotReference

A *SlotReference* represents the usage of a *Slot* in the context of PartUsage or PartOccurrence.

General Information

Base Classifier	AbstractSlotReference
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavityReference	cavityReference	0..*	1	Y	Specifies the CavityReferences used in the SlotReference.

7.11.19 Class SpliceTerminalRole

A SpliceTerminalRole defines the instance specific properties and relationships of a splice terminal.

General Information

Base Classifier	TerminalRole
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
spliceType	SpliceType	0..1	Specifies the type of splice (inline, end).
insulationState	InsulationState	0..1	Specifies the insulation state of the splice, in other words is electrically insulated or not.

7.11.20 Class TerminalReceptionReference

A *TerminalReceptionReference* is an instance of a *TerminalReception*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique *identification* of the *TerminalReceptionReference*. The *identification* is guaranteed to be unique within the *TerminalRole* (this might be for example a reception number).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

TerminalReception	terminalReception	1	0..*	N	References the *TerminalReception* that is instanced by this *TerminalReceptionReference.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
MatingDetail	0..*	secondTerminalReception	1	N	References the second terminal reception that is mated.
TerminalRole	1	terminalReceptionReference	0..*	Y	Specifies the *TerminalReceptionReferences* of this *TerminalRole.*
MatingDetail	0..*	firstTerminalReception	1	N	References the first terminal reception that is mated.
CavityMountingDetail	0..*	terminalReceptionReference	1	N	References the TerminalReception that is used for the detailed description of the cavity mounting.

7.11.21 Class TerminalRole

A TerminalRole defines the instance specific properties and relationships of a terminal.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
sealState	SealState	0..1	Specifies if this instance of a terminal should be sealed (waterproof). This applies for example to SpliceTerminals.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireReceptionReference	wireReceptionReference	0..*	1	Y	Specifies the *WireReceptionReferences* of this *TerminalRole.*
TerminalSpecification	terminalSpecification	1	0..*	N	References the *TerminalSpecification* that is instanced by this *TerminalRole.*
ComponentPort	componentPort	0..1	0..*	N	References the ComponentPort that is realized by the referenced Terminal (OccurrenceOrUsage with TerminalRole). KBLFRM-341
TerminalReceptionReference	terminalReceptionReference	0..*	1	Y	Specifies the *TerminalReceptionReferences* of this *TerminalRole.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ContactPoint	0..*	mountedTerminal	0..1	N	References the terminal that is used for contacting defined by the ContactPoint.

CavityReference	0..1	integratedTerminalRole	0..1	Y	Contains the terminal role if the cavity has an integrated terminal (e.g. an IDC).
MatingPoint	0..*	secondTerminalRole	1	N	References the second terminal that is mated.
PinComponentReference	0..1	terminalRole	0..1	Y	References the TerminalRole of PinComponentReference. This is required to specify a Mating for EEComponents with other EEComponents or a Harness. (KBLFRM-401)
MatingPoint	0..*	firstTerminalRole	1	N	References the first terminal that is mated.

7.11.22 Class WireElementReference

A WireElementReference represents the usage of a WireElement in the context of a PartUsage or PartOccurrence. For contacting purposes, a WireElementReference has WireEnds. KBLFRM-384

General Information

Base Classifier	RoutableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the WireElementReference. The identification is guaranteed to be unique within the WireRole.
unconnected	Boolean	0..1	If this attribute is *true*, the *WireElementReference* is not connected (unused) on purpose. This can be the case for example if a multi core is used, but not all cores are necessary in a specific situation. However, for all *WireElements* defined in the *WireSpecification* a corresponding *WireElementReference* shall exist. This attribute can be used to mark these unused cores explicitly.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Connection	connection	0..1	0..*	N	References the Connection that is realized by the referenced WireElement (WireElementReference). KBLFRM-341
WireLength	wireLength	0..*	1	Y	Specifies the different length of a wire.
WireElement	referencedWireElement	1	0..*	N	References the WireElement that is represented by the WireElementReference.
Signal	signal	0..1	0..*	N	References the signal that is transmitted by the WireElementReference.
ConnectionGroup	connectionGroup	0..1		N	References the *ConnectionGroup* that is realized by this *WireElementReference.* This applies normally to *WireElementReference* that have *subWireElements*.
WireEnd	wireEnd	0..*	1	Y	Specifies the ends of the WireElementReference for contacting purposes.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireGrouping	0..*	relatedWireElementReference	0..*	N	References the concrete wire elements (*WireElementReference*) that are grouped by the WireGrouping.
WireRole	1	wireElementReference	0..*	Y	Specifies the WireElementReferences used in the WireRole. For multi core wires more than one WireElementReference is needed.

7.11.23 Class WireEnd

A WireEnd is the end of a wire. This class mainly needed for the definition of a contacting. As a wire can be contacted on more than two ends (e.g. IDC) the WireEnd has a position on the wire.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the WireEnd. The identification is guaranteed to be unique within the WireElementReference.
positionOnWire	Double	1	Specifies the position of the WireEnd on the wire. This must be a value between 0 and 1.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConnectionEnd	connectionEnd	0..1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireMounting	0..*	referencedWireEnd	1..*	N	References the wire ends that are used for the wire mounting. The minimum cardinality is one, because a wire mounting without wire end makes no sense. The maximum cardinality is * in order to support multi crimps.
WireMountingDetail	0..*	referencedWireEnd	1..*	N	References the WireEnds that are mounted to referenced WireReception. A cardinality of more than one is allowed in order support parallel connectors, where multiple wire ends are placed on one side of the connector (one wire reception) and the other wire ends are placed on the other side of the connector (the other wire reception).
WireElementReference	1	wireEnd	0..*	Y	Specifies the ends of the WireElementReference for contacting purposes.

7.11.24 Class WireEndAccessoryRole

A *WireEndAccessoryRole* defines the instance specific properties and relationships of a *WireEndAccessory*.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireEndAccessory Specification	wireEndAccessory Specification	1		N	References the *WireEndAccessorySpecification* that is instanced by this *WireEndAccessoryRole.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireMounting	0..*	wireEndAccessory	0..*	N	

7.11.25 Class WireGrouping

A *WireGrouping* is the definition of a multi core wire in its usage. The elements of a *WireGrouping* are well defined wires (e.g. with a part number). The grouping itself is only created in its concrete usage. The most common use case is the individual definition of unshielded twisted pair wires without creating the full combinatorics of every possible core / insulation / twist combination in a part master data system (and by this creating part numbers for all of them). However, there are other use cases as well.

A *WireGrouping* groups the *relatedWireElementReferences* on an equal level containedWireGroupings are on a lower level. So, in order to create something like a shielded twisted pair, one *WireGrouping* "A" that references two *WireElementReferences* is required to represent the twisted pair and another *WireGrouping "*B" that contains *WireGrouping* "A" and references the "shield wire element".

The referenced *WireGroupSpecification* defines the handling of the *WireGrouping* during its assembly (e.g. twist).

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the WireGrouping. The identification is guaranteed to be unique within the WireGroupingSpecification.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

WireElementReference	relatedWireElementReference	0..*	0..*	N	References the concrete wire elements (*WireElementReference*) that are grouped by the WireGrouping.
WireGroupSpecification	wireGroupSpecification	0..1	0..*	N	References the *WireGroupSpecification* that applies to the *WireGrouping*.
ConnectionGroup	connectionGroup	0..1		N	References the *ConnectionGroup* that is realized by this *WireGrouping.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireGrouping	0..1	containedWireGroupings	0..*	Y	References the *WireGroupings* that are contained in this *WireGrouping*.
WireGroupingSpecification	0..1	wireGrouping	1..*	Y	Specifies the WireGroupings described by the WireGroupingSpecification.

7.11.26 Class WireGroupingSpecification

Specification for the description of WireGroupings.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireGrouping	wireGrouping	1..*	0..1	Y	Specifies the WireGroupings described by the WireGroupingSpecification.

7.11.27 Class WireLength

Defines the length of a WireElementReference. A WireElementReference can have multiple lengths of different types but must not have more than one length of the same type.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
lengthType	WireLengthType	1	The type of the length. Possible values are defined in an open enumeration (see WireLengthType).
lengthValue	NumericalValue	1	The length value for the type.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireElementReference	1	wireLength	0..*	Y	Specifies the different length of a wire.

7.11.28 Class WireReceptionReference

A *WireReceptionReference* is an instance of a *WireReception*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique *identification* of the *WireReceptionReference*. The *identification* is guaranteed to be unique within the *TerminalRole* (this might be for example a reception number).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireReception	wireReception	1	0..*	N	References the *WireReception* that is instanced by this *WireReceptionReference.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireMountingDetail	0..*	contactedWireReception	1	N	References the WireReception that is used for the WireMounting.
TerminalRole	1	wireReceptionReference	0..*	Y	Specifies the *WireReceptionReferences* of this *TerminalRole.*

7.11.29 Class WireRole

A WireRole defines the instance specific properties and relationships of a wire.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

WireElementReference	wireElementReference	0..*	1	Y	Specifies the WireElementReferences used in the WireRole. For multi core wires more than one WireElementReference is needed.
WireSpecification	wireSpecification	1	0..*	N	References the *WireSpecification* that is instanced by this *WireRole.*

7.11.30 Enumeration InsulationState

Enumeration for the definition of the insulation state of the splice.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Insulated		The splice is electrically insulated.
Uninsulated		The splice is not electrically insulated.

7.11.31 Enumeration SealState

Enumeration for the definition of the sealing state of the splice.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Unsealed		The splice is not waterproof.
Sealed		The splice is waterproof.

7.11.32 Enumeration SpliceType

Enumeration for the definition of type of splice.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
InlineSplice		An inlineSplice is a splice where wires from both sides are connected.
EndSplice		An endSplice is a splice where wires only from one side are connected.

7.11.33 Enumeration WireLengthType

Specifies possible values for the *lengthType* of *WireLength*.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
DMU		The length of the wire is calculated from the sum of the lengths of the neutral axes of the corresponding segments in the DMU model.
Drawing		The length is rounded length from the DMU model (shown on the drawing), without any add-ons
Contract		The agreed length for any negotiations and calculations.
Production		The cutting length for the used in production environments.

7.12 Module instancing_non_electrical_parts

7.12.1 Class CableDuctRole

A *CableDuctRole* defines the instance specific properties and relationships of a cable duct.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CableDuctSpecification	cableDuctSpecification	1	0..*	N	References the *CableDuctSpecification* that is instanced by this *CableDuctRole*.

7.12.2 Class CableLeadThroughReference

A *CableLeadThroughReference* is the instance of a CableLeadThrough. It can define a set of plugs or seals that are used together with it. Plugs are used if no wire is present, seals are used together with a wire.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the CableLeadThroughReference. The identification is guaranteed to be unique within the GrommetRole.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CableLeadThrough		0..1		N	
CavitySealRole	usedSeals	0..*	0..*	N	
CavityPlugRole	usedPlugs	0..*	0..*	N	References the plugs that are used with this CableLeadThroughReference. This association might be a 150% selection.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
GrommetRole	1	cableLeadThroughReference	0..*	Y	

7.12.3 Class CableTieRole

A *CableTieRole* defines the instance specific properties and relationships of a cable tie.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CableTieSpecification	cableTieSpecification	1	0..*	N	

7.12.4 Class FerriteRole

A *FerriteRole* defines the instance specific properties and relationships of a ferrite.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
numberOfWindings	Integer		Defines the of windings that the wires are wound around the ferrite.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

FerriteSpecification	ferriteSpecification	1		N	

7.12.5 Class FixingRole

A FixingRole defines the instance specific properties and relationships of a fixing.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
FixingSpecification	fixingSpecification	1	0..*	N	References the *FixingSpecification* that is instanced by this *FixingRole.*

7.12.6 Class GrommetRole

A GrommetRole defines the instance specific properties and relationships of a grommet.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CableLeadThroughReference	cableLeadThroughReference	0..*	1	Y	
GrommetSpecification	grommetSpecification	1	0..*	N	References the *GrommetSpecification* that is instanced by this *GrommetRole.*

7.12.7 Class TapeRole

Specific *WireProtectionRole* for instances of *TapeSpecification.*

General Information

Base Classifier	WireProtectionRole
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

tapeOverlap	NumericalValue	0..1	Specifies the amount of overlap two rounds of taping around a segment have, as an absolute value. A negative value means, that there is a gap between two rounds.
tapeOverlapRelative	Double	0..1	Specifies the amount of overlap two rounds of taping around a segment have, as a relative value. A negative value means, that there is a gap between two rounds. Values are defined as a factor, not as a percentage. Values can be between 1.0 and negative "infinity". **Examples:** An overlap of 1.0 defines that the second round is placed exactly on top of the first one. An overlap of 0.5 specifies that one half of the second round is on top of the first round (50% overlapping), a value of 0 specifies, that there is no overlap, but also no gap. A value of -2.0 specifies that there is a gap twice the width of the tape between two rounds.
tapingDirection	TapingDirection	0..1	Specifies the direction of the taping.
gradient	ValueWithUnit	0..1	Specifies the gradient of the taping.
windingType	WindingType	0..1	Defines the type of the tape's winding (see *WindingType*).
windingFirmness	WindingFirmness	0..1	Defines the firmness of the tape's winding (see WindingFirmness).

7.12.8 Class WireProtectionRole

A WireProtectionRole defines the instance specific properties and relationships of a wire protection. This is a general-purpose role for instances of all types of *WireProtectionSpecifications* that do not have specific instance attributes. For *TapeSpecifications* the more specific *TapeRole* shall be used.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
protectionLength	NumericalValue	0..1	Specifies the length of the protection.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
WireProtectionSpecification	wireProtectionSpecification	1	0..*	N	References the *WireProtectionSpecification* that is instanced by this *WireProtectionRole.*

7.12.9 Enumeration TapingDirection

Defines the direction in relation to the start & end-Location of the corresponding placement. If no TapingDirection is defined it is arbitrary.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		Comment
FromStart		
FromEnd		

7.12.10 Enumeration WindingFirmness

Defines the firmness with which a tape is applied to a segment.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Limp		The taping is applied limp (loose taping).
Tight		The taping is applied tight.

7.12.11 Enumeration WindingType

Defines the type of taping.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
OnSpace		*OnSpace* describes a taping with gaps between the rounds (German: Luecke)
Spare		*Spare* describes a taping with larger gaps between the rounds (German: Spar)
Overlap		*Overlap* describes a taping where each round overlaps the preceeding round.
DoubleOverlap		*DoubleOverlap* describes a taping where each round overlaps the preceeding round and the taping is done twice (forward and backwards).
Spiral		*Spiral* defines a taping with the least overlapping (German: Spiral).
Longitudinal		*Longitudinal* defines a taping with where the tape is folded around a segment.

7.13 Module instructions

7.13.1 Class DocumentBasedInstruction

A DocumentBasedInstruction is an Instruction to a SheetOrChapter in a DocumentVersion or to a complete DocumentVersion.

General Information

Base Classifier	Instruction

Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
DocumentVersion	referencedDocument	1	0..*	N	References the DocumentVersion that is used as an Instruction.
SheetOrChapter	referencedSheetOrChapter	0..1	0..*	N	References the SheetOrChapter that is used as an Instruction.

7.13.2 Class FileBasedInstruction

A FileBasedInstruction is an Instruction that references a file packaged (VEC-Package) together with a VEC-file. Such a file can be for example an image.

General Information

Base Classifier	Instruction
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
fileName	String	1	The name of the file as it appears in the package, including the folder structure (fully qualified name).
lastModified	Date	0..1	The last modified timestamp of the file.
dataFormat	String	0..1	The dataFormat specifies the format of the FileReference. This could be for example the mime-type of the external file.

7.13.3 Class Instruction

Abstract super class to specify different types of instructions. Possible instructions are text, file or document based.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
OccurrenceOrUsage	0..1	installationInstruction	0..*	Y	Room to specify InstallationInstruction(s) for the OccurrenceOrUsage.

7.13.4 Class TextBasedInstruction

A TextBasedInstruction is used to specify human readable instructions in text.

General Information

Base Classifier	Instruction
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
installationInstruction	LocalizedString	1..*	Specifies the instruction in a human readable form. Different languages are possible.

7.14 Module local_geometry

7.14.1 Class LocalGeometrySpecification

A *LocalGeometrySpecification* is responsible to create a relationship between the 3D model of a component (e.g. a connector, a cable duct or a fixing) and entities of the VEC. The 'Local' in the name refers to the fact that all definitions within this specification are 'local' to the 3D model of a specific component (a component in a library, not in a specific usage).

Specifically, it defines a transformation to transform the *BoundingBox* of a component into to coordinate system of the component and it defines the positions of *Placement-* and *MeasurementPoints* in this coordinate system.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Transformation3D	boundingBoxPositioning	0..1	0..1	Y	The transformation that defines the positioning of the bounding box in coordinate system of the component.
Unit	baseUnit	1		N	The *Unit* in which all coordinates (e.g. cartesian points) are defined. Shall be a unit of length (e.g. millimetre).
CartesianPoint3D	cartesianPoint	0..*	0..1	Y	All *CartesianPoint3Ds* that are used in this *LocalGeometrySpecification.* All *CartesianPoint3Ds* are defined in relation to the coordinate system of the component.
LocalPosition	positions	0..*	1	Y	All position defined by this *LocalGeometrySpecification.*

7.14.2 Class LocalPosition

A *LocalPosition* defines the position of a relevant point of component within the coordinate system of the component.

General Information

Base Classifier	

Applied Stereotype	
Is Abstract	true

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CartesianPoint3D	cartesianPoint	1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
LocalGeometrySpecification	1	positions	0..*	Y	All position defined by this *LocalGeometrySpecification.*

7.14.3 Class MeasurePointPosition

Defines the position of a MeasurementPoint within the coordinate system of the component. MeasurementPoint onto which dimension can be defined.

General Information

Base Classifier	LocalPosition
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
MeasurementPoint	measurementPoint	1	0..*	N	

7.14.4 Class PlacementPointPosition

Defines the position of a PlacementPoint within the coordinate system of the component. PlacementPoints are points where a component is attached to a *TopologySegment.* Therefor a *PlacementPointPosition* can define a tangent vector (in the coordinate system of the component) for a segment that is connected to the *PlacementPoint.*

General Information

Base Classifier	LocalPosition
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
tangent	CartesianVector3D	0..1	A vector in the direction of the tangent on the *TopologySegment* that is attached to the *PlacementPoint* represented by this instance. The vector is defined in the coordinate system of the 3D model of the component.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlacementPoint	placementPoint	1	0..*	N	

7.15 Module mapping

7.15.1 Class CavityMapping

Defines the mapping of two cavities contained Slot A & B of the containing SlotMapping-object.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identificationA	String	1	The identification of the Cavity on side A
identificationB	String	1	The identification of the Cavity on side B

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SlotMapping	1	cavityMapping	0..*	Y	

7.15.2 Class Mapping

The Mapping defines the concrete mapping two parts aliased as A & B. For performance reasons the roles PartSideA and PartSideB are abbreviated to A & B.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
SlotMapping	slotMapping	0..*	1	Y	
PartVersion	A	1	0..*	N	
PartVersion	B	1	0..*	N	

Incoming Relations

Other End	This End	General

Type	Mult	Role	Mult	Agg	Comment
MappingSpecification	1	mapping	0..*	Y	

7.15.3 Class MappingSpecification

Specification for the description of a PinMapping. A PinMapping allows the mapping between the Cavities of two independent ConnectorHousingSpecifications. This is for example needed to determine the Mating for an Inliner or for the Mating between an EE-Component and a ConnectorHousing. The PinMapping is a part master data.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Mapping	mapping	0..*	1	Y	

7.15.4 Class SlotMapping

Defines the mapping of two slots contained PartVersion A & B of the containing Mapping-object.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identificationA	String	1	The identification of the Slot on side A
identificationB	String	1	The identification of the Slot on side B

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CavityMapping	cavityMapping	0..*	1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Mapping	1	slotMapping	0..*	Y	

7.16 Module modules

7.16.1 Class ModuleFamily

A ModuleFamily is a mechanism to group mutually exclusive modules. This could be for example something like "audio equipment", "diesel engine".

In other words, a module family groups different variants of the same basic feature. In a real car configuration only one member of the family can participate. For the example module family "audio equipment" the members may be named: "Basic Audio Equipment", "Standard Audio Equipment", "Premium / High End Audio Equipment".

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the ModuleFamily. The identification is guaranteed to be unique within the ModuleFamilySpecification. For all VEC-documents a ModuleFamily-instance can be trusted to be the same if the ModuleFamilySpecification-instance is the same and the identification of the ModuleFamily is the same.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the ModuleFamily.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartWithSubComponentsRole	moduleInFamily	1..*	0..*	N	References the Modules that belong to the ModuleFamily.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ModuleFamilySpecification	1	moduleFamily	1..*	Y	Specifies the ModuleFamilies defined in the ModuleFamilySpecification.

7.16.2 Class ModuleFamilySpecification

Specification for the description of module families (see ModuleFamily).

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End	This	General

Type	Role	Mult	Mult	Agg	Comment
ModuleFamily	moduleFamily	1..*	1	Y	Specifies the ModuleFamilies defined in the ModuleFamilySpecification.

7.16.3 Class ModuleList

A ModuleList is a mechanism to control additional / completion PartOccurrences. This means for a car configuration, if at least one of the modules in the list participates in the configuration, the "completionComponent" participates, too.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the ModuleList. The identification is guaranteed to be unique within the ModuleListSpecification. For all VEC-documents a ModuleList-instance can be trusted to be the same if the ModuleListSpecification-instance is the same and the identification of the ModuleList is the same.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the ModuleList.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartWithSubComponentsRole	moduleInList	1..*	0..*	N	References the Modules that belong to the ModuleList. If any of the referenced Modules participates in a configuration the completion components participate, too.
OccurrenceOrUsage	completionComponents	1..*	0..*	N	References the components that are used as completion, if any of the Modules in the ModuleList appears in a configuration.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ModuleListSpecification	1	moduleListConfiguration	1..*	Y	Specifies the ModuleLists defined in the ModuleListSpecification.

7.16.4 Class ModuleListSpecification

Specification for the description of module lists (see ModuleList).

General Information

Base Classifier	Specification
Applied Stereotype	

Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ModuleList	moduleListConfiguration	1..*	1	Y	Specifies the ModuleLists defined in the ModuleListSpecification.

7.17 Module net

7.17.1 Class Net

A Net is an undirected link between *NetworkPorts*. It defines that the *NetworkPorts* are related to each other with the *Net*.

A *Net* is normally an instance of a *NetType.* E.g. if "CAN-BUS" is defined as a *NetType* typical *Nets* would be "BODY-CAN", "AUDIO-CAN".

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the Net. The identification is guaranteed to be unique within the NetSpecification.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the Net.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
NetType	netType	0..1		N	
NetworkPort	networkPort	1..*	1	N	References the NetworkPorts that are connected by the Net.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
NetSpecification	1	net	0..*	Y	Specifies the Nets defined in the NetSpecification.
Connection	0..*	net	0..1	N	References the Net that is realized by the Connection.
NetGroup	0..1	net	2..*	N	References the Nets that are grouped by the NetGroup.

7.17.2 Class NetGroup

Defines a logical grouping of specific *Nets*. For example, it can be used to identify all *Nets* of specific CAN domain, a function, a requirement level (e.g. Safety & Security).

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the NetGroup. The identification is guaranteed to be unique within the NetSpecification.
netGroupType	String	0..1	Specifies the type of the group.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the NetGroup.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Net	net	2..*	0..1	N	References the Nets that are grouped by the NetGroup.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
NetSpecification	1	netGroup	0..*	Y	Specifies the NetGroup defined in the NetSpecification.

7.17.3 Class NetSpecification

Specification for the description of electrological nets.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Net	net	0..*	1	Y	Specifies the Nets defined in the NetSpecification.
NetGroup	netGroup	0..*	1	Y	Specifies the NetGroup defined in the NetSpecification.
NetworkNode	networkNode	0..*	1	Y	Specifies the NetworkNodes defined in the NetSpecification.
NetType	netType	0..*	1	Y	

7.17.4 Class NetType

A *NetType* defines the different types of Nets used in the *NetSpecification*. The level of detail of the *NetTypes* for the can be process dependent.

A NetType could be just used to differentiate between conventional (analogue) communication and bus communication (digital), it can also already define the different types of digital communication (e.g. CAN, MOST, Ethernet).

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String		Specifies a unique identification of the NetType. The identification is guaranteed to be unique within the NetSpecification.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the *NetType*.
signalType	SignalType	0..1	
signalSubType	SignalSubType	0..1	
signalInformationType	SignalInformationType	0..1	
signalTransmissionMediumType	SignalTransmissionMediumType	0..1	Specifies the type of the transmission medium for signals of this type.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
NetworkPort	0..*	netType	0..1	N	
Net		netType	0..1	N	
Signal		netType	0..1	N	
NetSpecification	1	netType	0..*	Y	

7.17.5 Class NetworkNode

A NetworkNode is a representative for an actor in the electric system, e.g. an actuator, a sensor, an ECU

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the NetworkNode. The identification is guaranteed to be unique within the NetSpecification.
abbreviation	LocalizedString	0..*	Room for a short name of the NetworkNode.
networkNodeType	NetworkNodeType	0..1	Specifies the type of a NetworkNode. Common values are agreed as an *OpenEnumeration*.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the NetworkNode.
subType	NetworkNodeSubType	0..1	Specifies the sub type of a NetworkNode. The sub type allows a differentiation within a specific type. E.g. an actuator can be differentiated into lamps, speakers, motors.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
UsageNode	realizedUsageNode	0..1	0..*	N	References the *UsageNode* that is realized by this *NetworkNode*.
NetworkPort	port	0..*	1	Y	Specifies the NetworkPorts of a NetworkNode.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ComponentNode	0..*	networkNode	0..1	N	References the NetworkNode that is realized by the ComponentNode.
NetSpecification	1	networkNode	0..*	Y	Specifies the NetworkNodes defined in the NetSpecification.

7.17.6 Class NetworkPort

NetworkPort is the source or the receiver of a of a Net.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the NetworkPort. The identification is guaranteed to be unique within the NetSpecification.
signalDirection	SignalDirection	0..1	Specifies the direction of the signal on this NetworkPort.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the NetworkPort.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
NetType	netType	0..1	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ComponentPort	0..*	networkPort	0..1	N	References the NetworkPort that is realized by the ComponentPort.
NetworkNode	1	port	0..*	Y	Specifies the NetworkPorts of a NetworkNode.
Net	1	networkPort	1..*	N	References the NetworkPorts that are connected by the Net.

7.17.7 Enumeration NetworkNodeSubType

Defines agreed values for *NetworkNodeSubTypes*. Not all combinations of *NetworkNodeSubTypes* and *NetworkNodeTypes* are semantically correct (e.g. Lamp, Microphone, Speaker, Motor are all Actuators).

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Lamp		
Relay		
Fuse		
Microphone		
Speaker		
Motor		

7.17.8 Enumeration NetworkNodeType

Defines the common agreed values for the Types of a NetworkNode.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
ECU		Electronic Control Unit. An electronic control unit (ECU) is any embedded system in automotive electronics that controls one or more of the electrical systems or subsystems in a vehicle.
Sensor		A sensor is a device that detects changes to the environment and sends them as information to other devices (e.g. an ECU).

Actuator		An actuator is a device that is responsible for controlling or moving a mechanism.
CouplingDevice		A CouplingDevice is the (virtual) device that separates / connects two or more wiring harnesses. "Virtual" because it can be interpreted as a device / interface definition between the harnesses, where one harness behaves like an E/E component form the point of view of the other harness.
EnergyStorage		A device that stores energy in some kind of way (e.g. a battery).
Generator		A device that can generate energy.
PowerDistribution		A device that distributes power to other devices (e.g. a fuse box).
Switch		A "Switch" is a device that can change its internal connectivity in reaction to some external action (e.g. connect / disconnect some pins). A switch has no "active" logic (in contrast to an ECU).
OpenEnd		Defines that this NetworkNode is the end point for some unconnected nets that require wires and routings in the resulting harness (e.g. an antenna). A NetworkNode of this type is used whenever nets shall not be connected (on one side).
Ground		Defines that this NetworkNode is a grounding point.

7.17.9 Enumeration SignalDirection

Enumeration for the definition of SignalDirections.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
In		
Out		
InOut		

7.18 Module non_electrical_parts

7.18.1 Class BoltMountedFixingSpecification

Specification for fixings that are mounted onto a bolt. This means, the fixing itself has got a hole, which is mounted into a bolt.

General Information

Base Classifier	FixingSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
boltType	String	0..1	Specifies the type of the bolt on which the fixing can be mounted.
boltDiameter	NumericalValue	0..1	Specifies the diameter of the bolt on which the fixing can be mounted.
boltHeight	NumericalValue	0..1	Specifies the height of the bolt on which the fixing can be mounted.

7.18.2 Class CableDuctOutlet

Specifies one outlet of the cable duct.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies the identification of the Outlet. This must be unique within a CableDuctSpecification.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlacementPoint	placementPoint	0..1	0..*	N	Specifies the *PlacementPoint* that represents this *CableDuctOutlet* in a PlaceableElementSpecification.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CableDuctSpecification	1	outlet	0..*	Y	

7.18.3 Class CableDuctSpecification

Specification for cable ducts. A cable duct can have one or more outlets.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

CableDuctOutlet	outlet	0..*	1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CableDuctRole	0..*	cableDuctSpecification	1	N	References the *CableDuctSpecification* that is instanced by this *CableDuctRole*.

7.18.4 Class CableLeadThrough

A *CableLeadThrough* specifies a hole in the grommet through which wires can pass through the grommet. There can be different technical realizations of a lead through, e.g. it can be realized with a shrinking material or an additional seal. The properties of a *CableLeadThrough* are defined in the referenced *CableLeadThroughSpecification*.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies the identification of the CableLeadThrough. This must be unique within a GrommetSpecification.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlacementPoint	placementPoint	0..1	0..*	N	Specifies the *PlacementPoint* that represents this *CableLeadThrough* in a PlaceableElementSpecification.
CableLeadThroughSpecification	cableLeadThroughSpecification	0..1		N	References the *CableLeadThroughSpecification* that defines the technical properties of this *CableLeadThrough*.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CableLeadThroughReference			0..1	N	
GrommetSpecification	1	cableLeadThrough	0..*	Y	Specifies the CableLeadThroughs of the Grommet.

7.18.5 Class CableLeadThroughSpecification

A *CableLeadThrough* specifies a hole in the grommet through which wires can pass through the grommet. The *CableLeadThroughSpecification* defines the technical properties of a *CableLeadThrough* (or a set of similar ones).

General Information

Base Classifier	Specification

Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	CableLeadThroughType	0..1	Defines the type of a cable lead through. Standardized values are defined in an *OpenEnumeration*.
geometry	CableLeadThroughGeometry	0..1	Defines the geometry of a cable lead through in the sealing area. Standardized values are defined in an *OpenEnumeration*.
sealingDimension	Size	0..1	Specifies the dimension of the cable lead through in the sealing area.
minSegmentOutsideDiameter	NumericalValue	0..1	Specifies the minimum diameter a segment can have to fit through the cable lead through. This definition is necessary, since segments that are too small might cause movements and unacceptable torsion forces or they are not sealable.
maxSegmentOutsideDiameter	NumericalValue	0..1	Specifies the maximum diameter a segment can have to fit into the cable lead through.
sealable	Boolean	0..1	Specifies if the cable lead through is sealable.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
CableLeadThrough		cableLeadThroughSpecification	0..1	N	References the *CableLeadThroughSpecification* that defines the technical properties of this *CableLeadThrough.*

7.18.6 Class CableTieSpecification

Specification for the definition of cable ties.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
length	NumericalValue	0..1	
width	NumericalValue	0..1	
thickness	NumericalValue	0..1	
tensioningStrength	NumericalValue	0..1	

Incoming Relations

Other End	This End	General

Type	Mult	Role	Mult	Agg	Comment
CableTieRole	0..*	cableTieSpecification	1	N	

7.18.7 Class CorrugatedPipeSpecification

Specification for the definition of corrugated pipes.

General Information

Base Classifier	TubeSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
corrugationHeight	NumericalValue	0..1	Specifies the height of a corrugation of the pipe.
corrugationWidth	NumericalValue	0..1	Specifies the width of a corrugation of the pipe.
corrugationGradient	NumericalValue	0..1	Specifies the gradient of a corrugation of the pipe.

7.18.8 Class EdgeMountedFixingSpecification

Specification for fixings that are mounted onto an edge.

General Information

Base Classifier	FixingSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
edgeThickness	ValueRange	0..1	Defines a range of valid thicknesses, onto which the fixing can be mounted.

7.18.9 Class FerriteSpecification

Specification for the definition of ferrites.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

FerriteRole		ferriteSpecification	1	N	

7.18.10 Class FittingOutlet

Specifies one outlet of the fitting.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies the identification of the Outlet. This must be unique within a FittingSpecification.
innerDiameter	NumericalValue	0..1	Specifies the inner diameter of the outlet.
outerDiameter	NumericalValue	0..1	Specifies the outer diameter of the outlet.
nominalSize	String	0..1	Defines the nominal size of a tube. The nominal size is a name for the size of the tube that is somehow related to the inner diameter of the tube. However, it is not the inner diameter (e.g. "10.5").

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlacementPoint	placementPoint	0..1	0..*	N	Specifies the *PlacementPoint* that represents this *FittingOutlet* in a PlaceableElementSpecification.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
FittingSpecification	1	outlet	0..*	Y	

7.18.11 Class FittingSpecification

Specification for the definition of fittings. A fitting is a part that is used to connect pipes or tube. A fitting has a specific form (e.g. Y,T)

General Information

Base Classifier	WireProtectionSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
form	String	0..1	Specifies the form of the fitting (e.g. Y, T).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
FittingOutlet	outlet	0..*	1	Y	

7.18.12 Class FixingSpecification

Specification for the definition of fixings. A fixing is used to fix the harness in a certain position (e.g. at the car body, a seat, an ECU etc.). The FixingSpecification describes how the fixing is attached to the "non-harness" element. The attachment to harness is described by a PlaceableElementSpecification.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
offset	NumericalValue	0..1	
nominalSize	NumericalValue	0..1	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
FixingRole	0..*	fixingSpecification	1	N	References the *FixingSpecification* that is instanced by this *FixingRole.*

7.18.13 Class GrommetSpecification

Specification for the definition of grommets.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
hardness	NumericalValue	0..1	Specifies the hardness of the grommet.
holeDiameter	NumericalValue	0..1	Specifies the valid diameter of a hole into which the grommet fits.
plateThickness	ValueRange	0..1	Specifies valid the plate thickness at the hole into which the grommet fits.
grommetType	String	0..1	Specifies the type of the grommet.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
CableLeadThrough	cableLeadThrough	0..*	1	Y	Specifies the CableLeadThroughs of the Grommet.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
GrommetRole	0..*	grommetSpecification	1	N	References the *GrommetSpecification* that is instanced by this *GrommetRole.*

7.18.14 Class HoleMountedFixingSpecification

Specification for fixings that are mounted with a hole. This means, the fixing itself has got a hole, which is mounted onto a bolt.

General Information

Base Classifier	FixingSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
holeDiameter	NumericalValue	0..1	Specifies the diameter of the hole in which the fixing is mounted.
plateThickness	ValueRange	0..1	Specifies the thickness of the plate in which the hole is positioned.
holeType	String	0..1	Specifies the type of the hole in which the fixing can be mounted.
holeShape	String	0..1	

7.18.15 Class ShrinkableTubeSpecification

Specification of tubes that are shrinkable.

General Information

Base Classifier	TubeSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
shrinkingFactor	Double	0..1	Defines the factor of shrinking for the tube.
maximumLongitudinalShrinkage	NumericalValue	0..1	Defines the shrinkage in longitudinal direction.
resin	Material	0..*	Defines the material of the resin usable for this shrinkable tube.

waterAbsorbtion	NumericalValue	0..1	Defines the water absorption of the shrinkable tube specification.

7.18.16 Class StripeSpecification

Specifies a stripe which has fixed length and width. A stripe is a textile, foam or similar piece with fixed length & width that is wrapped around the harness.

General Information

Base Classifier	WireProtectionSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
length	NumericalValue	0..1	Specifies the length of the stripe.
segmentDiameter	ValueRange	0..1	Specifies the valid segment diameter range for which the stripe can be used.
width	NumericalValue	0..1	Specifies the width of the stripe.
thickness	NumericalValue	0..1	Specifies the thickness of the stripe (adhesive + backing).

7.18.17 Class TapeSpecification

Specification for the description of tapes.

General Information

Base Classifier	WireProtectionSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
backing	Material	0..*	Specifies the material of carrier of the tape, on which the adhesive material is applied.
adhesive	Material	0..*	Specifies the adhesive material of the tape.
width	NumericalValue	0..1	Specifies the width of the tape.
thickness	NumericalValue	0..1	Specifies the thickness of the tape (adhesive + backing).
coilCoreDiameter	NumericalValue	0..1	Specifies the inner diameter of the coil on which the tape is delivered.

7.18.18 Class TubeSpecification

Specifies tubes.

General Information

Base Classifier	WireProtectionSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
bendRadius	NumericalValue	0..1	Specifies the bend radius of the tube.
innerDiameter	NumericalValue	0..1	Defines the inner diameter of a tube. In the case of a shrinkable tube, it is the diameter of the tube in the unshrinked state.
wallThickness	NumericalValue	0..1	Specifies the thickness of the wall of the tube.
isSlit	Boolean	0..1	Specifies whether the tube is slit or not. The style of the slitting can be defined with the *slitStyle.* If a *slitStyle* is defined, it implies that *isSlit=true*.
slitStyle	TubeSlitStyle	0..1	Specifies the style of the slitting of the tube. If a *slitStyle* is defined, it implies that *isSlit=true*. This attribute is defined as an *OpenEnumeration.*
nominalSize	String	0..1	Defines the nominal size of a tube. The nominal size is a name for the size of the tube that is somehow related to the inner diameter of the tube. However, it is not the inner diameter (e.g. "10.5").
secondaryNominalSize	String	0..1	Defines the secondary nominal size of a tube. The nominal size is a name for the size of the tube that is somehow related to the inner diameter of the tube. However, it is not the inner diameter (e.g. "10.5"). The secondary nominal size shall only be used for two-parted tubes (see TubeSlitStyle = TwoParts). The secondary nominal size defines the size of the outer (larger) tube of a two-parted tube.
shape	TubeShape	0..1	Specifies the shape of the cross section of the tube. This attribute is defined as an *OpenEnumeration*
outerDiameter	NumericalValue	0..1	Specifies the outer diameter of the tube. The outer diameter of a tube shall only be used for circular tubes (shape = Circular). For other shapes, height and width shall be used.
height	NumericalValue	0..1	Specifies the height of the tube. If the height is defined, a width shall be defined, too. The height and width of a tube shall only be used for tubes that are not circular (shape != Circular). For circular shapes, the outside diameter shall be used.
width	NumericalValue	0..1	Specifies the width of the tube. If the width is defined, a height shall be defined, too. The height and width of a tube shall only be used for tubes that are not circular (shape != Circular). For circular shapes, the outside diameter shall be used.
length	NumericalValue	0..1	Specifies the length of the tube if it is a predefined value.

7.18.19 Class WireProtectionSpecification

Specification for the description of wire protections.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
soundDampingClass	SoundDampingClass	0..*	Specifies the class of sound damping. According to the VDA this is a value between A & E. KBLFRM-311

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
WireProtectionRole	0..*	wireProtectionSpecification	1	N	References the *WireProtectionSpecification* that is instanced by this *WireProtectionRole.*

7.18.20 Enumeration CableLeadThroughGeometry

Defines valid values for the geometry of a *CableLeadThrough* in the sealing area.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Square		
Circular		
Oval		

7.18.21 Enumeration CableLeadThroughType

Defines valid types for CableLeadThroughs.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
ShrinkingTube		
ShrinkingTubeWithSealingGlue		
SealingGlue		
Overmolded		

Foamed		
WithSealingComponent		Literal used when the CableLeadThrough is sealed with an additional sealing component (a "cavity seal").

7.18.22 Enumeration TubeShape

Defines valid shapes of the cross section of a tube.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Circular		The tube has circular cross section.
NonCircular		The tube has a cross section that is not circular.

7.18.23 Enumeration TubeSlitStyle

Defines valid types / styles of a slitted tube.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Slit		The tube has just a simple slit.
SelfClosing		The slit of the tube is self-closing.
Closable		The slit can be closed manually.
Overlapping		The edges of the slit are overlapping.
TwoParts		Two-parted tubes consist of inner tube and an outer tube (normally defined as one *PartNumber)*. Both tubes are slit and are combined into each other during assembly, thus creating one closed tube.

7.19 Module part_structure

7.19.1 Class CompositionSpecification

The CompositionSpecificiation is used to define a set of occurrences required to describe unambiguously the design of a composite part. This does not have to be necessarily the same occurrences which are building the bill of material. Example: A company might want to regard an antenna cable as one part out of a bill of material perspective. However, at the same time it may be useful for the company to be able to describe the contacting of the antenna cable within the VEC. (see also PartStructureSpecification)

General Information

Base Classifier	Specification

Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartOccurrence	component	0..*	1	Y	Specifies the PartOccurrences defined in the CompositionSpecification.

7.19.2 Class PartOccurrence

A PartOccurrence is an instance of a component with a specified part number (PartVersion).

General Information

Base Classifier	OccurrenceOrUsage
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
isSecondaryAlternative	Boolean	0..1	If a PartUsage is realized by more than one PartOccurrence it is possible to specify which one is the preferred. (see KBLFRM-264)

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartUsage	realizedPartUsage	0..*	0..*	N	References the PartUsages that are realized by the PartOccurrence.
PartVersion	part	0..1	0..*	N	References the PartVersion that is instantiated by this PartOccurrence.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PartOccurrence	0..*	instanciatedOccurrence	0..*	N	References the PartOccurrence which is instantiated by the PartOccurrence. This reference is for example needed in the case of usage of assemblies.
CompositionSpecification	1	component	0..*	Y	Specifies the PartOccurrences defined in the CompositionSpecification.
PartOccurrence	0..*	alternativeOccurrence	0..*	N	References the PartOccurrences that are an alternative for this PartOccurrence.

7.19.3 Class PartStructureSpecification

Specification for the description of a part structure. This specification defines the PartOccurrences that are in the bill of material of the described PartOrUsage. This is necessary, because it is possible, that for a definite description a Part more PartOccurrences are needed than the ones that are in the bill of material (see CompositionSpecification).

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
content	String	0..1	Specifies the type of content of the bill of material (e.g. module, harness complete set)

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
OccurrenceOrUsage	inBillOfMaterial	0..*	0..*	N	References the PartOccurrences that are building the bill of material of a composite part.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PartWithSubComponentsRole	0..*	partStructureSpecification	1	N	References the *PartStructureSpecification* that is instantiated by this *PartWithSubComponentsRole*.

7.20 Module part_usage

7.20.1 Class PartUsage

PartUsages shall be used for the specification of the elements on an electrical system wiring plan and for the specification of the elements on a pure geometry description. PartUsages shall more than ever be used in cases where it is necessary to describe a certain instance of a part or part group, possibly together with certain technical properties, but where it is at the same time yet not possible to define a concrete part number.

General Information

Base Classifier	OccurrenceOrUsage
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
primaryPartUsageType	PrimaryPartType	1	The primary type of the PartUsage defines the type of the described part (e.g. ConnectorHousing, Fixing, etc.) Since the VEC supports dual use parts (e.g. Fixing & WireProtection) there is no direct connection between the primaryPartUsageType and the allowed specifications for the description of a PartUsage.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

PartSubstitutionSpecification	partSubstitution	0..1		N	
PartOrUsageRelatedSpecification	partOrUsageRelatedSpecification	0..*	0..*	N	References the PartOrUsageRelatedSpecification(s) that describe the PartOrUsageRelatedSpecification. KBLFRM-399

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PartOccurrence	0..*	realizedPartUsage	0..*	N	References the PartUsages that are realized by the PartOccurrence.
PartUsageSpecification	1	partUsage	0..*	Y	Specifies the PartUsages defined by the PartUsageSpecification.
PartUsage	0..*	instanciatedUsage	0..*	N	

7.20.2 Class PartUsageSpecification

Specification for the definition of PartUsages.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartUsage	partUsage	0..*	1	Y	Specifies the PartUsages defined by the PartUsageSpecification.

7.21 Module pdm

7.21.1 Class Approval

Defines the approval of an ItemVersion. This consists of the StatusOfApproval and the Permissions issued for the approval.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the approval. The identification is guaranteed to be unique over all VEC-documents. Normally this would reference to a company specific approval number or something similar. KBLFRM-349

description	AbstractLocalizedString	0..*	Room for additional information about the item.
companyInScope	String	0..*	Room to specify for which companies the Approval is valid. It might be e.g. that an approved Item may only be delivered by some specific company.
status	StatusOfApproval	1	The approval level concerning approval status. Predefined are the values: NotYetApproved, Approved and Withdrawn. The status refers to the status of the *Approval*, not the status of the *ItemVersion.* So, e.g. withdrawn means the approval (with its corresponding level) has been withdrawn, not the *ItemVersion* itself.
levelOfApproval	LevelOfApproval	0..1	Relates to activities that are allowed with the ItemVersion after release for example building pilot tools or production tools. The levelOfApproval applies to the *ItemVersion* itself without further detailing or additional context. So, for example "Free" means, that from a component's perspective the corresponding *PartVersion* has satisfied all qualification procedures and can be used within its specified limits without restriction. The levels *Planned*, *Free*, *Invalid* refer to a single approval level. The levels *Develop* and *Restricted* refer to a category of approval levels in the lifecycle of an *ItemVersion* that all belong to the same phase but are highly company specific. For example, *Develop* approvals might be a "a start of construction approval" or a "building of prototypes or tools allowed approval". "Restricted" approvals might be "only for special purpose vehicles", "spare part only" or "residual parts may be used up". In these cases, the *additionalLevelInformation* can be used to provide further information (e.g. a company specific approval level).
additionalLevelInformation	String	0..1	Additional potentially company specific information about the level of approval (e.g. further detailing of a "Restricted" approval.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Permission	permission	0..*	1	Y	Specifies the permission issued with the approval.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ItemVersion	1	approval	0..*	Y	Specifies the approval information of the ItemVersion.

7.21.2 Class ChangeDescription

A ChangeDescription describes the implemented issues that are reason for the aggregating ItemVersion to be either an initial or successor version. A ChangeDescription can optionally define the person who has approved the change.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies an identifier for the ChangeDescription. This can be the ID of a change order or an ID which indicates model upgrading. (see KBLFRM-249)
label	String	0..1	Specifies the label of the change on a drawing for example. If more than one change is issued with one ItemVersion (e.g. change 0001, 0002), in some cases the different changes are labelled on the drawing (e.g. A, B, C).
description	AbstractLocalizedString	0..*	Specifies additional, human readable, information about the ChangeDescription.
approver	Person	0..1	Specifies the person who has approved the change.
changeDate	Date	0..1	Specifies the date when the change was performed.
responsibleDesigner	Person	0..1	Specifies the design engineer who is/was responsible to perform the change.
relatedChangeRequest	String	0..*	Specifies the identification of a corresponding change request. (see KBLFRM-249)

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
SheetOrChapter	0..1	changeDescription	0..*	Y	Specifies the change history of the SheetOrChapter.
ItemVersion	0..1	changeDescription	0..*	Y	Specifies the change history of the ItemVersion.

7.21.3 Class Contract

A Contract-instance describes the relationship between an ItemVersion-instance and a Company-instance additionally defining the role the company takes in reference to the ItemVersion.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	constant
Is Abstract	false

Attributes

Name	Type	Mult	Comment
companyName	String	1	Specifies the company which acts in the specified Role in the Contract Relationship.
contractRole	ContractRole	1	The role the company takes in reference to the associated ItemVersion. Predefined are the values: OEM, Supplier and Manufacturer.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
VecContent	1	contract	0..*	Y	Specifies the contracts used in the VEC-file.

ItemVersion	0..*	contract	0..*	N	References the contracts that apply to an ItemVersion.

7.21.4 Class CopyrightInformation

A CopyrightInformation-instance specifies copyright information for one or more Items.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	constant
Is Abstract	false

Attributes

Name	Type	Mult	Comment
copyrightNote	LocalizedString	1..*	An informal text which specifies copyright information.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
VecContent	0..*	standardCopyright Information	0..1	N	References the *CopyrightInformation* that is in effect for the complete content of this *VecContent*. It is applied to all *ItemVersions* that do not references their own individual *CopyrightInformation.*
ItemVersion	0..*	copyrightInformation	0..1	N	References the *CopyrightInformation* that is in effect for this *ItemVersion.* If no *CopyrightInformation* is referenced by the *ItemVersion*, the *CopyrightInformation* that is referenced by the *VecContent* (if defined) shall be considered as in effect for this *ItemVersion.*
VecContent	1	copyrightInformation	0..*	Y	Specifies the CopyrightInformation used in the VEC-file.

7.21.5 Class Creation

A Creation-instance provides additional information to the owning ItemVersion stating personal information on the creator and the creation date.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
creationDate	Date	1	Specifies the date when the associated ItemVersion was created. (see KBLFRM-241)
creator	Person	0..1	Specifies the person who has created the Item.

responsibleDesigner	Person	0..1	Specifies the person, which is the responsible designer for the ItemVersion at the point of creation.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ItemVersion	1	creation	0..1	Y	Specifies the information about the creation of the ItemVersion.

7.21.6 Class ItemEquivalence

Defines two or more ItemVersions to be equivalent out of the view of a certain company. The ItemEquivalence class will most likely be used by a company to express which PartNumber a certain PartVersion has in the context of other companies (same applies to *DocumentVersions*). However, for every other company separate ItemVersion-instances are needed as the statement of equivalence can be very subjective.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
companyName	String	1	Specifies the company which states the ItemEquivalence.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ItemVersion	item	2..*	0..*	N	References all ItemVersion that are considered to be equivalent by the ItemEquivalence. A single *ItemEquivalence* shall only reference *ItemVersions* of the same class (either *DocumentVersions* or *PartVersions*).

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
DocumentVersion	1	itemEquivalence	0..*	Y	Specifies ItemEquivalances defined by the DocumentVersion.

7.21.7 Class ItemHistoryEntry

An ItemHistroyEntry defines the direct relationship between ItemVersions in the terms of predecessor and successor. There are two possible types of relationships between ItemVersions, derivation and sequence. Derivation means for example for parts, that the successor version is a new part developed on the base of the predecessor version. Sequence means the successor version is an improvement of the predecessor version. By the combination of more than one ItemHistoryEntry a linear sequence of ItemVersions can be represented.

General Information

Base Classifier	ExtendableElement

Applied Stereotype	constant
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	HistoryEntryType	1	Specifies the type of relationship between the ItemVersions.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ItemVersion	successorVersion	1	0..*	N	References the ItemVersion that is the successor in the ItemHistoryEntry.
ItemVersion	predecessorVersion	1	0..*	N	References the ItemVersion that is the predecessor in the ItemHistoryEntry.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
VecContent	1	itemHistoryEntry	0..*	Y	Specifies the ItemVersionHistoryEntries for ItemVersions contained in the VEC-file.

7.21.8 Class Permission

Describes an act of acceptance together with information about the responsible person, department and company who directly provoked the approval level and status as described in the referenced Approval-instance. (see KBLFRM-229)

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
permission	TypeOfPermission	1	Specifies the type of permission. Predefined values are: Seen, Checked, Released.
permissionDate	Date	0..1	Specifies the date when the permission was stated.
permitter	Person	0..1	Specifies the person who was involved in the approval process giving a certain Permission.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Approval	1	permission	0..*	Y	Specifies the permission issued with the approval.

7.21.9 Class Person

Specifies all relevant data of a person.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
companyName	String	0..1	Specifies the name of the company the person belongs to.
department	String	0..1	Specifies the department the person belongs to.
firstName	String	0..1	Specifies the person's first name.
lastName	String	1	Specifies the person's last name.
phoneNumber	String	0..*	Specifies the person's phone number.
emailAddress	String	0..*	Specifies the person's email address.
aliasId	AliasIdentification	0..*	Specifies identifiers for the *Person* in different contexts.

7.21.10 Class Project

Specifies a certain vehicle project. Instances of this class are assumed to be constant. Thus, this is located directly under VEC-root element.

A vehicle project can be some abstract or concrete node in the product structure, addressed by the car classification levels.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	constant
Is Abstract	false

Attributes

Name	Type	Mult	Comment
carClassificationLevel2	String	0..1	Provides a classification according to "CC8 Recommended Practices Specification and Configuration, Product Structures". Car classification is the identification of a set of similar cars to be offered to the market. Level 2 stands for "Technical information / platform" and reflects the level of a product class in a BoM system which represents a main technical product base (e.g. project, platform, engineering series etc.). In some cases, this level carries a complete BoM ("Maximum BoM") for a project, platform, engineering series etc. This level is in some cases called technical documentation.
carClassificationLevel3	String	0..1	Provides a classification according to "CC8 Recommended Practices Specification and Configuration, Product Structures". Car

			classification is the identification of a set of similar cars to be offered to the market. Level 3 stands for "Configuration information / product family" where all variant control mechanisms are attached.
carClassificationLevel4	String	0..1	Provides a classification according to "CC8 Recommended Practices Specification and Configuration, Product Structures". Car classification is the identification of a set of similar cars to be offered to the market. Level 4 stands for "Furthest pre-configured abstract product class" and represents the furthest configured class of a product, which is not yet a real product. E.g. this could be a complete vehicle, engine, gearbox etc. which has not been evaluated against customer special choices or an abstract vehicle, engine, gear-box etc. which could become a real one after the associated BoM is evaluated. The purpose of this level of a product class instance is in any case to reflect that level of product class of a BoM system which leads to the individual BoM for a single product.
identification	String	0..1	Specifies the development order number (car or engine project)
description	AbstractLocalizedString	0..*	Room for additional information about the item.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ApplicationConstraint		project	0..*	N	Defines the projects for which the application constraint applies.
UsageNode	0..*	usedInProject	0..*	N	Specifies the *Projects* in which the *UsageNode* can be used.
PartVersion	0..*	project	0..1	N	References the project that develops the PartVersion.
UsageConstraint	0..*	project	0..*	N	References the *Projects* to which the *UsageConstraint* applies. This means the described PartVersion is allowed / denied in the referenced UsageConstraint.
VecContent	1	project	0..*	Y	Specifies the Projects used in the VEC-file.

7.21.11 Enumeration ContractRole

Enumeration for the definition of roles a contractor has in a contract.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Oem		
Supplier		
Manufacturer		

7.21.12 Enumeration HistoryEntryType

Enumeration for the definition of the type of relationship represented by an ItemHistoryEntry. (see KBLFRM-271)

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Derivation		
Sequence		

7.21.13 Enumeration LevelOfApproval

The LevelOfApproval standardizes the approval levels most engineering processes have in common. See *Approval.levelOfApproval* for more details.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Planned		The *ItemVersion* has been created.
Develop		An approval during the development process. There might process specific different approvals out of this category. See Approval.additionalLevelInformation for more information about the details.
Free		The ItemVersion can be used without restriction.
Restricted		The Usage of the ItemVersion is restricted in some way. See additionalLevelInformation for more details.
Invalid		The *ItemVersion* shall not be used anymore.

7.21.14 Enumeration StatusOfApproval

Enumeration for the definition of the state of an approval.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
NotYetApproved		
Approved		
Withdrawn		

7.21.15 Enumeration TypeOfPermission

Enumeration for the definition of the type of a permission.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Seen		
Checked		
Released		

7.22 Module physical_information

7.22.1 Class BoundingBox

The bounding box is used to define a cuboid (box) that can contain a described part completely. Therefore, it is a simplified representation of the bounding volume and represents a definition of the maximum volume occupied by the part.

It is valid to use the *BoundingBox* to describe the dimensions of a component, even if not all dimensions are known (e.g. only length and width). However, it must be possible to transform such a partial bounding box into a complete bounding box by adding the missing dimensions.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
x	NumericalValue	0..1	Defines the extent of the bounding box in the direction of x (length).
y	NumericalValue	0..1	Defines the extent of the bounding box in the direction of y (width).
z	NumericalValue	0..1	Defines the extent of the bounding box in the direction of *z* (height).

Incoming Relations

Other End		This End		General	
Type	Mult	Role	Mult	Agg	Comment
GeneralTechnicalPartSpecification		boundingBox	0..1	Y	Defines the bounding box of the part.

7.22.2 Class Color

Specifies a color value. A color is always defined by a key value. What color is meant by this key value is defined by a standard reference system (e.g. RAL).

For example, if a RAL color should be expressed in the terms of the VEC the *referenceSystem* would be "RAL", the *key* would be the RAL number defined by the standard (e.g. "1003" for signal yellow).

Attributes of the type Color normally have the multiplicity [0..*]. This means that such an attribute can have a value for different referenceSystems (e.g. RAL, RGB,...). It must not have multiple values for the same ReferenceSystem.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
key	String	1	The key of the color in the corresponding color reference system.
referenceSystem	ColorReferenceSystem	1	The identification of the color reference system, which is defining possible values and the semantic of color keys. (see KBLFRM-315). For common color reference systems the literals defined in the open enumeration *ColorReferenceSystem* shall be used.
description	LocalizedString	0..*	On optional human readable description of the color (e.g. the name).

7.22.3 Class CompositeUnit

Defines a unit as a composition of other units. The composition is done by multiplying the different other units. By this way combined units like kg/m can be formed.

General Information

Base Classifier	Unit
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Unit	factors	1..*		N	References the *Units* that are used as factors to create the *CompositeUnit.*

7.22.4 Class CustomUnit

A CustomUnit can be used to define "FreeText"-Units. Custom units must not be used for units that can be expressed by any of the other subclasses of Unit. Custom units are only allowed if a unit is needed that cannot be handled by any of the other classes.

General Information

Base Classifier	Unit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	A unique identification of the custom unit.

7.22.5 Class IECUnit

The IECUnit class can define quantities in the terms of the IEC-Unit-System by specifying the corresponding IEC prefix (optional) and an IEC unit name.

General Information

Base Classifier	Unit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
iecUnitName	IECUnitName	1	Specifies the name of the IEC unit.
iecPrefix	IECPrefix	0..1	Specifies the prefix of the IEC unit.

7.22.6 Class ImperialUnit

The ImperialUnit class can define quantities in the terms of the Imperial-Unit-System by specifying the corresponding Imperial unit name.

General Information

Base Classifier	Unit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
imperialUnitName	ImperialUnitName	1	Specifies the name of the imperial unit.

7.22.7 Class MassInformation

Allows the definition of mass information. Attributes of the type MassInformation normally have the multiplicity [0..*]. This means that such an attribute can have mass values for different determinationTypes and valueSources. It must not have multiple values for the same determinationType and valueSource.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

determinationType	ValueDetermination	0..1	Specifies the determination type of the mass information.
value	NumericalValue	1	Specifies the mass as numerical value.
valueSource	MassInformationSource	0..1	The *valueSource* defines in an OpenEnumeration the source from which the *MassInformation* has been retrieved. This information, in combination with the *ValueDetermination* gives a hint about the reliability of the *MassInformation*

7.22.8 Class Material

Allows the definition of material information. Attributes of the type Material normally have the multiplicity [0..*]. This means that such an attribute can have material values for different *referenceSystems*. It must not have multiple values for the same *referenceSystems*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
key	String	1	The key of the material in the corresponding material reference system.
referenceSystem	String	1	The identification of the material reference system, which is defining possible values and the semantic of material keys.
description	LocalizedString	0..*	On optional human readable description of the material (e.g. the name).

7.22.9 Class NumericalValue

A quantity expressed with a numerical value and a unit.

General Information

Base Classifier	ValueWithUnit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
valueComponent	Double	1	Specifies the value of the numerical value.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Tolerance	tolerance	0..1	0..1	Y	Specifies the tolerance for the dimension.

7.22.10 Class OtherUnit

The OtherUnit class can be used to define a unit, which is necessary in the context of data exchange but not contained in the standard systems (e.g. Piece).

General Information

Base Classifier	Unit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
otherUnitName	OtherUnitName	1	Specifies the name of the unit.

7.22.11 Class RobustnessProperties

Allows the definition of robustness properties. Robustness of component is specified as a level of robustness against a specific influence (e.g. oil, water, UV-light). The influence is specified by the *class* and the level is specified by the *classKey*. Valid robustness classes and keys are specified by the reference system. Attributes of the type RobustnessProperties normally have the multiplicity [0..*]. This means that such an attribute can have RobustnessProperties entries for different *classReferenceSystems* and *classes*. It must not have multiple entries for the same *class* and *classReferenceSystem*.

Note: Most reference systems just define one class or at least some of the possible classes, but not all (e.g. the ISO 20653 defines "Solid Particle Protection" and "Liquid Ingress Protection", whereas the ISO 6722 defines "Ambient Temperature" among others).

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
class	RobustnessClass	1	Specifies the identifier of a robustness class defined by the robustness class reference system. Robustness classes are for example: oil, petrol, UV, water. Specific known and used classes are defined in an open enumeration.
classKey	String	0..1	Specifies a key for the robustness level defined in the specified robustness class (e.g. A, B, C).
classReferenceSystem	RobustnessClassReferenceSystem	1	The identification of the robustness class reference system, which is defining possible values and the semantic of robustness classes and robustness class keys. Specific known and used reference systems are defined in an open enumeration.
hasRobustness	Boolean	1	Specifies if the described element has a robustness in the specified robustness class. (see KBLFRM-260)
description	LocalizedString	0..*	On optional human readable description of the robustness (e.g. the name).

7.22.12 Class SIUnit

The SIUnit class can define quantities in the terms of the SI-Unit-System by specifying the corresponding SI prefix (optional) and a SI unit name. The usage of SI units must be the preferred way of expressing units, since these units can be easily translated into other SI units.

General Information

Base Classifier	Unit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
siUnitName	SiUnitName	1	Specifies the name of SI unit (e.g. metre, second,...)
siPrefix	SiPrefix	0..1	Specifies the prefix of the SI unit (e.g. milli, centi, mirco,...)

7.22.13 Class Size

Defines the size of an element by width & height. Per definition is width > height.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
width	NumericalValue	1	The width of the element.
height	NumericalValue	1	The height of the element.

7.22.14 Class SoundDampingClass

Allows the definition of a sound damping class. The sound damping class of a component is specified as a level of sound damping. The level is specified by the *classKey*. Valid keys are specified by the *referenceSystem*. Attributes of the type *SoundDampingClass* normally have the multiplicity [0..*]. This means that such an attribute can have *SoundDampingClass* entries for different *referenceSystems*. It must not have multiple entries for the same *referenceSystem*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

classKey	String	1	Specifies a key for the level defined in the sound damping class (e.g. A, B, C).
referenceSystem	String	1	The identification of the sound damping class reference system, which is defining possible values and the semantic of sound damping keys.

7.22.15 Class TemperatureInformation

Defines the temperature information for a general technical part. It is necessary to define this in an external class and not as an attribute, since a part can have multiple different temperature information e.g. operating temperature, storage temperature, processing temperature, environment temperature. An additional constraint is that one GeneralTechnicalPartSpecification can own multiple TemperatureInformations but must not have more than one TemperatureInformations of the same temperatureType.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
temperatureRange	ValueRange	0..1	Specifies the allowed temperature range for this type of temperature.
temperatureType	String	0..1	The type of a TemperatureInformation. Possible values are: EnvironmentTemperature, OperationTemperature, StorageTemperature, ProcessingTemperature.

7.22.16 Class Tolerance

Enables the specification of value ranges which can be tolerated.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
lowerBoundary	Double	1	Specifies the lower boundary for the tolerance.
upperBoundary	Double	1	Specifies the upper boundary for the tolerance.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Dimension	0..1	tolerance	0..1	Y	
DefaultDimension	0..1	toleranceIndication	1	Y	

NumericalValue	0..1	tolerance	0..1	Y	Specifies the tolerance for the dimension.

7.22.17 Class Unit

A precisely specified quantity in terms of which the magnitudes of other quantities of the same kind can be stated. The different systems to define units are represented by the subclasses of this class (e.g. SIUnit, ImperialUnit).

General Information

Base Classifier	
Applied Stereotype	constant
Is Abstract	true

Attributes

Name	Type	Mult	Comment
exponent	Integer	0..1	Defines the exponent with which this unit instance should be used. In order to define square meters for example, the SIUnit "metre" with an exponent of 2 will be used. If no exponent is defined it is equivalent to the value 1.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Dimension	0..*	unitComponent	1	N	
ValueWithUnit	0..*	unitComponent	1	N	References the unit of the value.
VecContent	1	unit	0..*	Y	Specifies the Units used in the VEC-file.
BuildingBlockSpecification2D	0..*	baseUnit	1	N	
BuildingBlockSpecification3D	0..*	baseUnit	1	N	
LocalGeometrySpecification		baseUnit	1	N	The *Unit* in which all coordinates (e.g. cartesian points) are defined. Shall be a unit of length (e.g. millimetre).
CompositeUnit		factors	1..*	N	References the *Units* that are used as factors to create the *CompositeUnit.*

7.22.18 Class USUnit

The USUnit class can define quantities in the terms of the US-Unit-System by specifying the corresponding US unit name. The US Unit System is quite similar to the imperial unit system; however, some units are defined slightly different.

General Information

Base Classifier	Unit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
usUnitName	USUnitName	1	Specifies the name of the *USUnit*.

7.22.19 Class ValueRange

A pair of numerical values representing a value range.

General Information

Base Classifier	ValueWithUnit
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
minimum	Double	1	Lower bound of the value range.
maximum	Double	1	Upper bound of the value range.

7.22.20 Class ValueWithUnit

Abstract class either for a single numerical measure or a range of numerical measures with upper, lower, or upper and lower bounds.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	true

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Unit	unitComponent	1	0..*	N	References the unit of the value.

7.22.21 Enumeration ColorReferenceSystem

Defines the literals that shall be used for specific color reference systems.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
IEC 60757		The literal shall be used for the IEC 60757 "Electrotechnical engineering; code for designation of colours"
RAL		

RGB		

7.22.22 Enumeration IECPrefix

Enumeration for the definition of IEC unit prefixes.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Yobi		
Zebi		
Exbi		
Pebi		
Tebi		
Gibi		
Mebi		
Kibi		

7.22.23 Enumeration IECUnitName

Enumeration for the definition of IEC unit names.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Bit		
Byte		

7.22.24 Enumeration ImperialUnitName

Enumeration for the definition of imperial unit names.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**

Thou		
Inch		
Foot		
Yard		
Chain		
Furlong		
Mile		
League		
FluidOunce		
Gill		
Pint		
Quart		
Gallon		
Grain		
Drachm		
Ounce		
Pound		
Stone		
Quarter		
HundredWeight		
Ton		
Perch		
Rood		
Acre		

7.22.25 Enumeration MassInformationSource

Defines possible sources for *MassInformations*.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Prototype		A *MassInformation* with this *valueSource* has been determined based on a preproduction part.
Series		A *MassInformation* with this *valueSource* has been determined based on a serial production part.
IMDS		A *MassInformation* with this *valueSource* has been retrieved from the *IMDS* (see http://www.mdsystem.com).

7.22.26 Enumeration OtherUnitName

Enumeration for the definition of other unit names.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Pi		
Piece		

7.22.27 Enumeration RobustnessClass

Defines the literals that shall be used for specific robustness classes.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
AmbientTemperature		
SolidParticleProtection		
LiquidIngressProtection		
ImpactProtection		

7.22.28 Enumeration RobustnessClassReferenceSystem

Defines the literals that shall be used for specific robustness class reference systems.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
ISO 20653		The literal that shall be used for the ISO 20653 "Road vehicles - Degrees of protection (IP code) - Protection of electrical equipment against foreign objects, water and access".
ISO 6722		The literal that shall be used for the ISO 6722 "Road vehicles -- 60 V and 600 V single-core cables".
LV 112		The literal that shall be used for the LV 112.
LV 214		OEM Connector Test Specification
LV 215		OEM Specification for High Voltage contacting.
LV 312		Protective systems for wiring harnesses in motor vehicles Hoses

7.22.29 Enumeration SiPrefix

Enumeration for the definition of SI unit prefixes.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		Comment
Yotta		
Zetta		
Exa		
Peta		
Tera		
Giga		
Mega		
Kilo		
Hecto		
Deca		
Deci		
Centi		
Milli		
Micro		

Nano		
Pico		
Femto		
Atto		
Zepto		
Yocto		

7.22.30 Enumeration SiUnitName

Enumeration for the definition of SI unit names. In difference to the SI-System gram is used as literal for the measurement of mass, instead of kilogram. Since a unit is formed with prefix and unit name gram would have to be defined as milli kilogram otherwise.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Metre		
Gram		
Second		
Ampere		
Kelvin		
Mole		
Candela		
Radian		
Steradian		
Hertz		
Newton		
Pascal		
Joule		
Watt		
Coulomb		

Volt		
Farad		
Ohm		
Siemens		
Weber		
Tesla		
Henry		
DegreeCelsius		
Lumen		
Lux		
Becquerel		
Gray		
Sievert		
Katal		

7.22.31 Enumeration USUnitName

Enumeration for the definition of US unit names.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Thou		
Inch		
Foot		
Yard		
Chain		
Furlong		
Mile		
League		

FluidOunce		
Gill		
Pint		
Quart		
Gallon		
Grain		
Drachm		
Ounce		
Pound		
Stone		
Quarter		
HundredWeight		
Ton		
Perch		
Rood		
Acre		
AWG		American wire gauge (AWG), also known as the Brown & Sharpe wire gauge, is a standardized wire gauge system used in the United States and Canada.

7.22.32 Enumeration ValueDetermination

Enumeration for the definition of a value determination. (see KBLFRM-316)

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Calculated		The value is determined by a calculation algorithm.
Measured		The value is determined by an exact measurement.
Estimated		The value is estimated by a person.

7.23 Module pin_wire_mapping

7.23.1 Class PinWireMappingPoint

The *PinWireMappingPoint* creates a single variance free mapping between a *ContactPoint* and a *PinComponentReference* within a *PinWireMappingSpecification* (more details there).

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinComponentReference	pinComponentReference	1		N	
ContactPoint	contactPoint	1		N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PinWireMappingSpecification				Y	

7.23.2 Class PinWireMappingSpecification

A *PinWireMappingSpecification* can be used create **variance free** mappings between a wire (represented by the contact point) and the pin of an E/E component. This is a possibility to create a shortcut in the model between a wire and its connected E/E-component (e.g. a fuse) that might be only indirectly connected to a wire (e.g. via a fuse and relay carrier). This is a relevant information for e.g. the validation of fusing. See the Pin Wire Mapping Diagram for more details.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PinWireMappingPoint				Y	

7.24 Module placeable_element

7.24.1 Class MeasurementPoint

Defines a reference point on a component that can be used to apply a Dimension. This is normally a significant point of the component e.g. an edge, a hole, a bolt or something similar.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the MeasurementPoint. The identification is guaranteed to be unique within a component.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
MeasurePointPosition	0..*	measurementPoint	1	N	
MeasurementPointReference	0..*	measurementPoint	1	N	References the *MeasurementPoint* that is instanced by this *MeasurementPointReference.*
PlaceableElementSpecification	1	measurementPoint	0..*	Y	Specifies the *MeasurementPoints* of a *PlaceableElement*.

7.24.2 Class PlaceableElementSpecification

Specification for the general aspects of a component that are enabling the component to be placed in a topology. All components that should have the ability to be placed on a certain position in the topology must have a PlaceableElementSpecification

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
validPlacementTypes	PlacementType	1..2	Defines the *PlacementTypes* that are valid for this element.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlacementPoint	placementPoint	0..*	1	Y	Specifies the *PlacementPoints* of a *PlaceableElementSpecification*.
MeasurementPoint	measurementPoint	0..*	1	Y	Specifies the *MeasurementPoints* of a *PlaceableElement*.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

PlaceableElement Role	0..*	placeableElement Specification	1	N	References the *PlaceableElementSpecification* that is instanced by this *PlaceableElementRole.*

7.24.3 Class PlacementPoint

Defines a point on the component which can be placed explicitly in the topology (e.g. opening in a grommet, the entry point of a connector housing). If a PlacementPoint requires further specification for a certain component type (e.g. the entry point of a connector housing), this is done by an element specific for the component type, which related to the placement point.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the PlacementPoint. The identification is guaranteed to be unique within a component.
segmentDiameter	ValueRange	0..1	Specifies the valid segment diameter range for which this *PlacementPoint* is suitable.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PlaceableElement Specification	1	placementPoint	0..*	Y	Specifies the *PlacementPoints* of a *PlaceableElementSpecification*.
PlacementPointReference	0..*	placementPoint	1	N	References the *PlacementPoint* that is instanced by this *PlacementPointReference.*
PlacementPointPosition	0..*	placementPoint	1	N	
CableLeadThrough	0..*	placementPoint	0..1	N	Specifies the *PlacementPoint* that represents this *CableLeadThrough* in a PlaceableElementSpecification.
WireReception	0..*	placementPoint	0..1	N	Specifies the *PlacementPoint* that represents this *WireReception* in a PlaceableElementSpecification.
FittingOutlet	0..*	placementPoint	0..1	N	Specifies the *PlacementPoint* that represents this *FittingOutlet* in a PlaceableElementSpecification.
SegmentConnectionPoint	0..*	placementPoint	0..1	N	Specifies the *PlacementPoint* that represents this *SegmentConnectionPoint* in a *PlaceableElementSpecification.*
CableDuctOutlet	0..*	placementPoint	0..1	N	Specifies the *PlacementPoint* that represents this *CableDuctOutlet* in a PlaceableElementSpecification.

7.24.4 Enumeration PlacementType

Defines the type for which a *PlaceableElement* can be used.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		Comment
OnWay		*OnWay* refers to an *OnWayPlacement*.
OnPoint		*OnPoint* refers to an *OnPointPlacement*.

7.25 Module placement

7.25.1 Class DefaultDimension

A *DefaultDimension* defines a tolerance value that shall be applied to a part, if no explicit tolerance value has been defined.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
dimensionValueRange	ValueRange	1	dimensionValueRange defines the magnitude of measurements for which the tolerance applies (e.g. length from 500mm - 1500mm).
dimensionType	String	1	The *dimensionType* defines to which measurements this *DefaultDimension* applies (e.g. segment lengths, wire lengths).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Tolerance	toleranceIndication	1	0..1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
DefaultDimensionSpecification				Y	

7.25.2 Class DefaultDimensionSpecification

A *DefaultDimensionSpecification* defines tolerances that shall be applied to a part, if no explicit tolerance value has been defined.

General Information

Base Classifier	Specification
Applied Stereotype	

Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
DefaultDimension				Y	

7.25.3 Class Dimension

A Dimension defines a measurement for the distance between two Locations. An acceptable tolerance can be specified additionally. If the Locations are not located on adjacent topology-elements it is possible to specify a Path for the dimension along which the measurement must be taken.

The value for the Dimension is not specified as NumericalValue (which can define a Tolerance as well). This is because the "valueComponent" of the NumericalValue is mandatory. For Dimensions it shall be optional since there are scenarios where the dimension only specifies a Tolerance for a distance defined by the topology (segment length + placement information).

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the Dimension. The identification is guaranteed to be unique within the containing PlacementSpecification.
valueComponent	Double	0..1	Defines the value of the dimension. This value can be null, if it shall be calculated and only a tolerance shall be defined.
valueCalculated	Boolean	0..1	Defines if the value of the *Dimension* was calculated (e.g. the sum of segment lengths in the topology) or if it was defined manually.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
DimensionAnchor	referenceAnchor	1	0..*	N	References the location that is used as reference point for the dimensioning (e.g. the location of a fixing as this relates to a fixpoint of the body in white). See KBLFRM-329 and KBLFRM-391.
Path	path	0..1	0..1	Y	Specifies a path in the topology along which the dimension is defined.
DimensionAnchor	dimensionAnchor	1	0..*	N	References the location that is used as dimension point for the dimensioning (e.g. the entry point of a bundle into a connector housing). See KBLFRM-329 and KBLFRM-391.
Location	definedLocations	0..2	1	Y	

Unit	unitComponent	1	0..*	N	
Tolerance	tolerance	0..1	0..1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PlacementSpecification	1	dimension	0..*	Y	Specifies the Dimensions defined by the PlacementSpecification.

7.25.4 Class DimensionAnchor

A *DimensionAnchor* represents an abstract anchor onto which a *Dimension* can be specified.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	true

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Dimension	0..*	referenceAnchor	1	N	References the location that is used as reference point for the dimensioning (e.g. the location of a fixing as this relates to a fixpoint of the body in white). See KBLFRM-329 and KBLFRM-391.
Dimension	0..*	dimensionAnchor	1	N	References the location that is used as dimension point for the dimensioning (e.g. the entry point of a bundle into a connector housing). See KBLFRM-329 and KBLFRM-391.

7.25.5 Class MeasurementPointReference

A *MeasurementPointReference* is the instance of a *MeasurementPoint* in the context of an *OccurrenceOrUsage*.

General Information

Base Classifier	DimensionAnchor
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the MeasurementPointReference. The identification is guaranteed to be unique within the containing PlaceableElementRole.

Outgoing Relations

Other End	This	General

Type	Role	Mult	Mult	Agg	Comment
MeasurementPoint	measurementPoint	1	0..*	N	References the *MeasurementPoint* that is instanced by this *MeasurementPointReference.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PlaceableElement Role	1	measurementPoint Reference	0..*	Y	

7.25.6 Class OnPointPlacement

An OnPointPlacement is a placement of an OccurrenceOrUsage that places it onto discrete points, in most cases one point. In some cases, it is necessary to place a component (with more than one reference point) onto multiple points (e.g. a cable channel).

General Information

Base Classifier	Placement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Location	location	1..*	0..1	Y	References the Locations where Placement places the reference points of the placed element.

7.25.7 Class OnWayPlacement

An OnWayPlacement places an OccurrenceOrUsage onto an area of the Topology (e.g. a tape or a tube). The area is defined by a startLocation and an endLocation. If startLocation and endLocation are not located on the same TopologySegment it is possible to specify a Path of TopologySegments over which the OnWayPlacement goes.

The names start- and endLocation are used to distinguish between the two ends. It does **not** indicate a direction as property of the product (e.g. for tapes).

General Information

Base Classifier	Placement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Path	path	0..1	0..1	Y	Specifies the topology path defining the way the OnWayPlacement takes in the topology.
Location	startLocation	1	0..1	Y	References the Location where OnWayPlacement starts.

Location	endLocation	1	0..1	Y	References the Location where OnWayPlacement ends.

7.25.8 Class PlaceableElementRole

A PlaceableElementRole defines the instance specific properties and relationships of a PlaceableElement.

General Information

Base Classifier	Role
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlaceableElement Specification	placeableElement Specification	1	0..*	N	References the *PlaceableElementSpecification* that is instanced by this *PlaceableElementRole.*
PlacementPointReference	placementPointReference	0..*	1	Y	
MeasurementPoint Reference	measurementPoint Reference	0..*	1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Placement	0..*	placedElement	1..*	N	References the PlaceableElementRoles placed by the Placement.

7.25.9 Class Placement

A placement defines the placement of a PlaceableElementRole onto a Topology. For the definition of the place on the Topology, Locations are used. A Placement can either be a placement on discrete points (OnPointPlacement) or on an area of the topology (OnWayPlacement).

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the Location. The identification is guaranteed to be unique within the PlacementSpecification.
type	String	0..1	Room to specify additional type information of the placement.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

PlaceableElement Role	placedElement	1..*	0..*	N	References the PlaceableElementRoles placed by the Placement.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Placement	0..*	isOnTopOf	0..*	N	Specifies constraints on ordering of Placements. All referenced Placements must be below (nearer to the center of the Segment) this Placement. (see KBLFRM-171)
PlacementSpecification	1	placement	0..*	Y	Specifies the Placements defined by the PlacementSpecification.

7.25.10 Class PlacementPointReference

A *PlacementPointReference* is the instance of a *PlacementPoint* in the context of an *OccurrenceOrUsage*.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the PlacementPointReference. The identification is guaranteed to be unique within the containing PlaceableElementRole.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlacementPoint	placementPoint	1	0..*	N	References the *PlacementPoint* that is instanced by this *PlacementPointReference.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Location	0..*	placedPlacementPoints	0..*	N	References the *PlacementPointReference* that is placed by this location.
PlaceableElement Role	1	placementPointReference	0..*	Y	

7.25.11 Class PlacementSpecification

Specification for the description of placements. *Placements* are used to connect *OccurrenceOrUsages* with a topology.

General Information

Base Classifier	Specification

Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Dimension	dimension	0..*	1	Y	Specifies the Dimensions defined by the PlacementSpecification.
Placement	placement	0..*	1	Y	Specifies the Placements defined by the PlacementSpecification.

7.26 Module requirements_conformance

7.26.1 Class RequirementsConformanceSpecification

A *RequirementsConformanceSpecification* can be used to express the conformance of a *PartVersion* (or a group of *PartVersions*). The *PartVersions* to which this *RequirementsConformanceSpecification* applies are expressed through the *describedPart* reference.

General Information

Base Classifier	PartOrUsageRelatedSpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
RequirementsConformanceStatement	conformanceStatement	0..*	1	Y	

7.26.2 Class RequirementsConformanceStatement

A *RequirementsConformanceStatement* states that the *PartVersions* referenced by the parent *RequirementsConformanceSpecification* satisfy or do not satisfy the requirements defined in the associated *DocumentVersion* (via the *requirementsSpecification* association).

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
satisfies	boolean		Defines if the *describedParts* satisfy (satisfy = true) or explicitly fail (satisfy = false) to conform with the *requirementsSpecification*.
description	LocalizedString	0..*	A free text description / additional information / comment for the *RequirementsConformanceStatement.*

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
DocumentVersion		1		N	References the *DocumentVersion* that contains the requirements to which a conformance statement shall be expressed.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
RequirementsConformanceSpecification	1	conformanceStatement	0..*	Y	

7.27 Module routing

7.27.1 Class Routing

A Routing is the assignment of a RoutableElement (Connection or WireElementReference) to a path in the topology.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the Routing. The identification is guaranteed to be unique within the RoutingSpecification.
specialRoutedComment	LocalizedString	0..1	Allows the specification of an explanation why this routing has been routed in a special way.
specialRouted	Boolean	0..1	Specifies that routing has been created in a special way. This means it has not been calculated in the standard way, because for some reason the result of the standard calculation has been inconvenient.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Path	path	1	0..1	Y	Specifies a Path on the topology that is used for the routing.
RoutableElement	routedElement	1	0..*	N	Specifies the Element that is routed.
TopologySegment	mandatorySegment	0..*	0..*	N	Specifies some constraints for the routing. If the path of the routing is recalculated the referenced segments must be visited.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

RoutingSpecification	1	routing	0..*	Y	Specifies the Routings described by the RoutingSpecification.

7.27.2 Class RoutingSpecification

Specification for the description of Routings.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Routing	routing	0..*	1	Y	Specifies the Routings described by the RoutingSpecification.

7.28 Module schematic

7.28.1 Class ComponentConnector

A *ComponentConnector* is a grouping of *ComponentPorts* and represents a logical abstraction of a connector of an *EEComponent*. When the *ComponentNode* is realized by an *EEComponentRole* the *ComponentConnector* will result in a *HousingComponent.*

It is permitted that a *ComponentConnector* is a 150% definition of connector.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the ComponentConnector. The identification is guaranteed to be unique within the ComponentNode.
description	AbstractLocalizedString	0..*	

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ComponentPort	componentPort	0..*	1	Y	Specifies the ComponentPorts of the ComponentConnector.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**

ComponentNode	1	componentConnector	0..*	Y	Specifies the ComponentConnectors of a ComponentNode.
HousingComponentReference	0..*	componentConnector	0..1	N	References the ComponentConnector that is realized by the referenced HousingComponentReference.

7.28.2 Class ComponentNode

A ComponentNode is a node where an electrological component is located. It is a representative for an element in the electric system, e.g. an actuator, a sensor, an ECU. In this way it is quite similar to a NetworkNode and may even reference the corresponding NetworkNode in this case. However, a ComponentNode is more likely to be used as a representative of an inliner or a splice. Moreover, a ComponentNode can define childNodes in order to describe its internal structure.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the ComponentNode. The identification is guaranteed to be unique within the ConnectionSpecification.
abbreviation	LocalizedString	0..*	Room for a short name of the ComponentNode.
componentNodeType	ComponentNodeType	0..1	Specifies the type of the ComponentNode.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the ComponentNode.
subType	ComponentNodeSubType	0..1	Specifies the sub type of a ComponentNode. The sub type allows a differentiation within a specific type. E.g. an actuator can be differentiated into lamps, speakers, motors.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ComponentConnector	componentConnector	0..*	1	Y	Specifies the ComponentConnectors of a ComponentNode.
UsageNode	realizedUsageNode	0..1	0..*	N	References the *UsageNode* that is realized by this *ComponentNode*.
NetworkNode	networkNode	0..1	0..*	N	References the NetworkNode that is realized by the ComponentNode.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConnectionSpecification	0..1	componentNode	0..*	Y	Specifies the ComponentNodes defined by the ConnectionSpecification.

ConnectorHousingRole	0..*	componentNode	0..1	N	References the ComponentNode that is realized by the referenced ConnectorHousing (OccurrenceOrUsage with ConnectorHousingRole). This can especially be relevant for inliners. KBLFRM-341.
EEComponentRole	0..*	componentNode	0..1	N	References the ComponentNode that is realized by the referenced EEComponent (OccurrenceOrUsage with EEComponentRole). KBLFRM-341
ComponentNode	0..1	childNode	0..*	Y	Specifies the ComponentNodes that are a child of this ComponentNode.

7.28.3 Class ComponentPort

Defines a port of ComponentNode. A ComponentPort is usually the realization of a NetworkPort. Electrological connections are defined between two or more ComponentPorts.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the ComponentPort. The identification is guaranteed to be unique within the ComponentConnector.
signalDirection	SignalDirection	0..1	Specifies the direction of the signal on this ComponentPort.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the ComponentPort.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Signal	signal	0..1	0..*	N	References the *Signal* that is associated with the *ComponentPort*.
NetworkPort	networkPort	0..1	0..*	N	References the NetworkPort that is realized by the ComponentPort.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConnectionEnd	0..*	connectedComponentPort	1	N	References the ComponentPort that is connected by the ConnectionEnd.
ComponentConnector	1	componentPort	0..*	Y	Specifies the ComponentPorts of the ComponentConnector.
TerminalRole	0..*	componentPort	0..1	N	References the ComponentPort that is realized by the referenced Terminal (OccurrenceOrUsage with TerminalRole).

					KBLFRM-341
CavityReference	0..*	componentPort	0..1	N	References the *ComponentPort* that is implemented by this *CavityReference*.

7.28.4 Class Connection

A Connection is an electrological connection between two or more ComponentPorts.

General Information

Base Classifier	RoutableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the Connection. The identification is guaranteed to be unique within the ConnectionSpecification.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the Connection.
installationInstruction	Instruction	0..*	Specifies installation instruction for the connection.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConnectionEnd	connectionEnd	2..*	1	Y	Specifies the ConnectionEnds of the Connection.
Net	net	0..1	0..*	N	References the Net that is realized by the Connection.
Signal	signal	0..1	0..*	N	References the signal that is transmitted by the connection.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
BridgeTerminalRole	0..*	connection	0..1	N	References the Connection that is realized by this BridgeTerminalRole.
WireElementReference	0..*	connection	0..1	N	References the Connection that is realized by the referenced WireElement (WireElementReference). KBLFRM-341
ConnectionGroup	0..*	connection	2..*	N	References the Connections that are grouped by the ConnectionGroup.
MatingDetail	0..*	connection	0..1	N	References the *Connection* that is realized by this *MatingPointDetail.* For example, when a connection is realized by directly plugging or screwing two E/E components together. The definition at level of the *MatingDetail* might be required if the *TerminalRole* of the MatingPoint carries multiple different potentials (e.g. Coax).

MatingPoint	0..*	connection	0..1	N	References the *Connection* that is realized by this *MatingPoint.* For example, when a connection is realized by directly plugging or screwing two E/E components together.
ConnectionSpecification	1	connection	0..*	Y	Specifies the Connection defined by the ConnectionSpecification.

7.28.5 Class ConnectionEnd

A connection end is the end of a Connection at a ComponentPort.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the ConnectionEnd. The identification is guaranteed to be unique within the ConnectionSpecification.
isExternalEnd	Boolean	1	Specifies if the ConnectionEnd is connected to the internal or the external side of the ComponentPort.
gender	String	0..1	Specifies if the ConnectionEnd is male or female. This may be e.g. important in case of an inliner.
installationInstruction	Instruction	0..*	Specifies installation instruction for the *ConnectionEnd*.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ComponentPort	connectedComponentPort	1	0..*	N	References the ComponentPort that is connected by the ConnectionEnd.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
Connection	1	connectionEnd	2..*	Y	Specifies the ConnectionEnds of the Connection.
WireEnd	0..*	connectionEnd	0..1	N	

7.28.6 Class ConnectionGroup

A ConnectionGroup references two or more Connections expressing that the physical realization of the referenced Connection shall be somehow grouped e.g. twisted. For complex structures a ConnectionGroup can specify subgroups.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	

Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	0..1	Specifies a unique identification of the ConnectionGroup. The identification is guaranteed to be unique within the ConnectionSpecification.
connectionGroupType	ConnectionGroupType	1	Specifies the type of the connectionGroup, valid literals are defined in the open enumeration *ConnectionGroupType*.
installationInstruction	Instruction	0..*	Specifies additional InstallationInstructions for the ConnectionGroup.
description	AbstractLocalizedString	0..*	Room for additional, human readable information about the ConnectionGroup.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Connection	connection	2..*	0..*	N	References the Connections that are grouped by the ConnectionGroup.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConnectionSpecification	0..1	connectionGroup	0..*	Y	Specifies the ConnectionGroup defined by the ConnectionSpecification.
WireElementReference		connectionGroup	0..1	N	References the *ConnectionGroup* that is realized by this *WireElementReference.* This applies normally to *WireElementReference* that have *subWireElements*.
WireGrouping		connectionGroup	0..1	N	References the *ConnectionGroup* that is realized by this *WireGrouping.*
ConnectionGroup	0..1	subGroup	0..*	Y	Specifies the ConnectionGroups that are a subgroup of this ConnectionGroup.

7.28.7 Class ConnectionSpecification

A ConnectionSpecification is used to define electrological connectivity.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ConnectionGroup	connectionGroup	0..*	0..1	Y	Specifies the ConnectionGroup defined by the ConnectionSpecification.

ComponentNode	componentNode	0..*	0..1	Y	Specifies the ComponentNodes defined by the ConnectionSpecification.
Connection	connection	0..*	1	Y	Specifies the Connection defined by the ConnectionSpecification.

7.28.8 Enumeration ComponentNodeSubType

Defines agreed values for *ComponentNodeSubTypes*. Not all combinations of *ComponentNodeSubTypes* and *ComponentNodeTypes* are semantically correct (e.g. Lamp, Microphone, Speaker, Motor are all Actuators).

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Lamp		
Relay		
Fuse		
Microphone		
Speaker		
Motor		

7.28.9 Enumeration ComponentNodeType

Defines the common agreed values for the types of a ComponentNode.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
ECU		Electronic Control Unit. An electronic control unit (ECU) is any embedded system in automotive electronics that controls one or more of the electrical systems or subsystems in a vehicle.
Sensor		A sensor is a device that detects changes to the environment and sends them as information to other devices (e.g. an ECU).
Actuator		An actuator is a device that is responsible for controlling or moving a mechanism.
CouplingDevice		A CouplingDevice is the (virtual) device that separates / connects two or more wiring harnesses. "Virtual" because it can be interpreted as a device / interface definition between the harnesses, where one harness behaves like an E/E component form the point of view of the other harness.
EnergyStorage		A device that stores energy in some kind of way (e.g. a battery).
Generator		A device that can generate energy.
PowerDistribution		A device that distributes power to other devices (e.g. a fuse box).

Switch		A "Switch" is a device that can change its internal connectivity in reaction to some external action (e.g. connect / disconnect some pins). A switch has no "active" logic (in contrast to an ECU).
Lamp		*Deprecated since VEC V1.2.* Use ComponentNodeType "Actuator" instead and ComponentSubType *Lamp*.
Relay		
Fuse		*Deprecated since VEC V1.2.* Use instead ComponentSubType Fuse.
Ground		Defines that this ComponentNode is a grounding point.
OpenEnd		Defines that this ComponentNode is the end point for some unconnected connections that require wires and routings in the resulting harness (e.g. an antenna). A component node of this type is used whenever connections shall not be connected (on one side), regardless if it is an individual wire or a core of a multi core wire.
OpenLink		Defines that this ComponentNode is an OpenLink. In the description of partial systems, it can be necessary to reference a ComponentNode that is not defined in the scope of the partial system (e.g. vehicle infrastructure like power, ground, bus systems). OpenLinks must be resolved and replaced by a determined ComponentNode when a partial system is integrated into a vehicle system.

7.28.10 Enumeration ConnectionGroupType

Defines the type of physical realization that is expressed by a *ConnectionGroup.*

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Twisted		
Shielded		
Coaxial		

7.29 Module signal

7.29.1 Class Signal

Specifies a signal.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the Signal. The identification is guaranteed to be unique within the SignalSpecification.
signalName	String	0..1	Name of the Signal, which is not guaranteed to be unique and is normally function oriented.
clampName	String	0..1	Specifies the name of the clamp e.g. KL15, KL30, KL31.
aliasId	AliasIdentification	0..*	Specifies additional identifiers for the *Signal*.
description	AbstractLocalizedString	0..*	A human readable description of the signal.
signalInformationType	SignalInformationType	0..1	Specifies the information type of the signal.
signalTransmissionMediumType	SignalTransmissionMediumType	0..1	Specifies the type of the transmission medium for the signal.
signalForm	SignalForm	0..1	Specifies the form of the signal.
signalCurve	SignalCurve	0..1	Specifies the curve of the signal.
signalType	SignalType	0..1	
signalSubType	SignalSubType	0..1	
currentType	CurrentType	0..1	
nominalVoltage	NominalVoltage	0..1	
dataRate	NumericalValue	0..1	Defines the data rate of the signal. This applies only to signals with *signalType = 'information*' and *signalInformationType = 'digital*'. For the numerical value, an appropriate IECUnit combination shall be used (e.g. GBit / Second).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
InsulationSpecification	recommendedInsulationSpecification	0..1		N	Defines a recommended Specification for the insulation (e.g. the color) that implements this signal.
NetType	netType	0..1		N	
ConductorSpecification	recommendedConductorSpecification	0..1		N	Defines a recommended Specification for the cores that implement this signal.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
PinComponentBehavior	0..*	signal	0..1	N	Specifies the *Signal* associated with the pin in this behavior.
ComponentPort	0..*	signal	0..1	N	References the *Signal* that is associated with the *ComponentPort*.

SignalSpecification	1	signal	0..*	Y	Specifies the signals.
WireElementReference	0..*	signal	0..1	N	References the signal that is transmitted by the WireElementReference.
Connection	0..*	signal	0..1	N	References the signal that is transmitted by the connection.

7.29.2 Class SignalSpecification

Specification for the definition of a list of valid signals.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Signal	signal	0..*	1	Y	Specifies the signals.

7.29.3 Enumeration CurrentType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
ACOnePhase		
ACTwoPhase		
ACThreePhase		
DC		

7.29.4 Enumeration NominalVoltage

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
12V		
24V		
48V		

HV		HV defines all voltages that are dangerous to health or life, regardless if alternating (AC) or direct current (DC). This literal should be used if the concrete class is not known, not defined or not specified.
HV1		HV Class 1 (200V DC)
HV2		HV Class 2 (300V DC)
HV3		HV Class 3 (600V DC)
HV4		HV Class 4 (900V DC)
HV5		HV Class 5 (1200V DC)

7.29.5 Enumeration SignalCurve

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
continuousSignal		
discreteSignal		

7.29.6 Enumeration SignalForm

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
sinusWave		
squareWave		

7.29.7 Enumeration SignalInformationType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
analog		
digital		

7.29.8 Enumeration SignalSubType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
CAN		Controller Area Network
LIN		Local Interconnect Network
FlexRay		
MOST		
Ethernet		
BroadR-Reach		
RTPGE		
APIX		Automotive Pixel Link
APIX2		
APIX3		
FBAS		
USB		Universal Serial Bus Version 1.X: The "USB" literal represents all USB 1.X Versions for USB 2.0 or USB 3.X the corresponding literals shall be used.
USB2		Universal Serial Bus Version 2.0
USB3.X		Universal Serial Bus Version 3.X
LVDS		low-voltage differential signalling
RGB		
BTLE		Bluetooth Low Energy
NFC		Near Field Communication
IEEE802.11		also: Wi-Fi, Wireless LAN
SignalGround		
PowerGround		

7.29.9 Enumeration SignalTransmissionMediumType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
electrical		
optical		
hydraulic		
pneumatic		
acoustic		
inductive		
radioTransmission		Signal transmission via electromagnetic waves (e.g. Wi-Fi, 4G, 5G)

7.29.10 Enumeration SignalType

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
information		
energy		
ground		

7.30 Module terminal_pairing

7.30.1 Class TerminalPairing

A *TerminalPairing* is a standard reference setup of exactly two terminals and a defined length of a core contacted to both terminals. The *TerminalPairing* defines physical properties that apply to this combination.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
contactResistance	NumericalValue	0..1	Specifies the resistance of the terminal pairing.
matingForce	NumericalValue	0..1	Specifies the joining force of the two terminals.
unmatingForce	NumericalValue	0..1	Specifies the force required to unmate the two terminals.

Outgoing Relations

Other End			This	General	
Type	Role	Mult	Mult	Agg	Comment
ConductorSpecification	referencedCoreSpecification	1	0..*	N	References the CoreSpecification that is used on both sides of the ContactSystem.
PartVersion	secondTerminal	1	0..*	N	References the second terminal of the TerminalPairing (first and second does not imply any specific order).
PartVersion	firstTerminal	1	0..*	N	References the first terminal of the TerminalPairing.

Incoming Relations

Other End		This End		General	
Type	Mult	Role	Mult	Agg	Comment
TerminalPairingSpecification	1	terminalPairing	0..*	Y	Specifies the TerminalPairings described by this TerminalPairingSpecification.

7.30.2 Class TerminalPairingSpecification

Specification for the definition of TerminalPairings.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	Role	Mult	Mult	Agg	Comment
TerminalPairing	terminalPairing	0..*	1	Y	Specifies the TerminalPairings described by this TerminalPairingSpecification.

7.31 Module topology

7.31.1 Class CartesianVector

Abstract super class for vectors, either 2D or 3D.

General Information

Base Classifier	
Applied Stereotype	

Is Abstract	true

7.31.2 Class GeometryNode

A GeometryNode is the geometric representation of a TopologyNode. A TopologyNode may be represented by more than one GeometryNodes, but only within different *BuildingBlockSpecification2D/3D.* That means, in a single BuildingBlockSpecification a *TopologyNode* shall only be represent once (or not).

A GeometryNode is either a GeometryNode2D or a GeometryNode3D.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the GeometryNode. The identification is guaranteed to be unique within the corresponding BuildingBlockSpecification. For all VEC-documents a GeometryNode-instance can be trusted to be the same if the BuildingBlockSpecification-instance is the same (see BuildingBlockSpecification) and the identification of the GeometryNode is the same.
aliasId	AliasIdentification	0..*	Specifies additional identifiers for the GeometryNode.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologyNode	referenceNode	0..1	0..*	N	References the TopologyNode that is represented by the GeometryNode.

7.31.3 Class GeometrySegment

A GeometrySegment is the geometric representation of a TopologySegment. A TopologySegment may be represented by more than one GeometrySegments, but only within different *BuildingBlockSpecification2D/3D.* That means, in a single BuildingBlockSpecification a *TopologySegment* shall only be represent once (or not).

Furthermore, the definition of the *GeometrySegment* shall be consistent to the definition in the *TopologySpecification.* That means, that a *GeometrySegment* shall have those *GeometryNodes* as start- & endNode that represent the *TopologyNodes* referenced from the corresponding *TopologySegment*.

A GeometrySegment is either a GeometrySegment2D or a GeometrySegment3D.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment

identification	String	1	Specifies a unique identification of the GeometrySegment. The identification is guaranteed to be unique within the corresponding BuildingBlockSpecification. For all VEC-documents a GeometrySegment-instance can be trusted to be the same if the BuildingBlockSpecification-instance is the same (see BuildingBlockSpecification) and the identification of the GeometrySegment is the same.
aliasId	AliasIdentification	0..*	Specifies additional identifiers for the GeometrySegment.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologySegment	referenceSegment	0..1	0..*	N	References the TopologySegment that is represented by the GeometrySegment.

7.31.4 Class Location

A Location specifies a distinct position on a topology. Locations can be used for the placement of components or for the definition of Dimensions.

General Information

Base Classifier	DimensionAnchor
Applied Stereotype	
Is Abstract	true

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the Location. The identification is guaranteed to be unique within the PlacementSpecification.
matchingId	String	0..1	Specifies an identification for matching the location with a reference point of component (e.g. a cable channel).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PlacementPointReference	placedPlacementPoints	0..*	0..*	N	References the *PlacementPointReference* that is placed by this location.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
OnPointPlacement	0..1	location	1..*	Y	References the Locations where Placement places the reference points of the placed element.
Dimension	1	definedLocations	0..2	Y	
NodeMapping	0..1	mappedPosition	1	Y	
ZoneCoverage	0..1	secondLocation	1	Y	

OnWayPlacement	0..1	startLocation	1	Y	References the Location where OnWayPlacement starts.
OnWayPlacement	0..1	endLocation	1	Y	References the Location where OnWayPlacement ends.
ZoneCoverage	0..1	firstLocation	1	Y	

7.31.5 Class NodeLocation

Specifies a TopologyNode as a Location.

General Information

Base Classifier	Location
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologyNode	referencedNode	1	0..*	N	References the TopologyNode on which the Location is located.

7.31.6 Class NodeMapping

Defines the relationship of an inner node to its outer topology. The relationship to the outer topology is expressed with a *Location*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologyNode	innerNode	1		N	
Location	mappedPosition	1	0..1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TopologyMapping Specification	1		0..*	Y	

7.31.7 Class Path

Describes a path in the topology. A *Path* is a continuous way through a topology without interruptions, defined by an ordered list of *TopologySegments*.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologySegment	segment	0..*	0..*	N	Specifies an ordered list of TopologySegments the routing goes through.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
OnWayPlacement	0..1	path	0..1	Y	Specifies the topology path defining the way the OnWayPlacement takes in the topology.
Dimension	0..1	path	0..1	Y	Specifies a path in the topology along which the dimension is defined.
Routing	0..1	path	1	Y	Specifies a Path on the topology that is used for the routing.
TopologyBendingRestriction	0..1	restrictedPath	1	Y	The path that defines the restricted way in the topology.
SegmentMapping	0..1	mappedPosition	1	Y	

7.31.8 Class SegmentCrossSectionArea

Specifies the cross-section area of a segment. For the data exchange the cross-section area is used and not the diameter, because the diameter is only a valid measure for circular segments. For circular segments, the diameter and the cross-section area can be translated into each other without the loss of information. Attributes of the type SegmentCrossSectionArea normally have the multiplicity [0..*]. This means that such an attribute can have SegmentCrossSectionArea entries for different crossSectionAreaTypes and valueDeterminations. It must not have multiple entries with the same crossSectionAreaType and valueDetermination.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
area	NumericalValue	1	Specifies the value of the cross-section area of the segment.
valueDetermination	ValueDetermination	1	Specifies the method for determination of the value.
crossSectionAreaType	SegmentCrossSectionAreaType	0..1	Specifies the type of the cross-section area of the segment. Different types are for example needed to differentiate between the

			designed space of a segment and the required space (e.g. calculated from the wires going through the segment). Attribute is defined as an *OpenEnumeration.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TopologySegment	1	crossSectionAreaInformation	0..*	Y	Specifies the different SegmentCrossSectionAreas of the TopologySegment.

7.31.9 Class SegmentLength

Specifies the length of a *TopologySegment*. The length of a *TopologySegment* is defined as the length of the centerline of the segment.

Attributes of the type SegmentLength normally have the multiplicity [0..*]. This means that such an attribute can have SegmentLength-entries for different classifications. It must not have multiple entries with the same classification.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
length	NumericalValue	1	Specifies the length of the TopologySegment.
classification	LengthClassification	1	Specifies the classification of the segment length.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TopologySegment	1	lengthInformation	0..*	Y	Specifies the different SegmentLengths of the TopologySegment.

7.31.10 Class SegmentLocation

Specifies a point on a TopologySegment as a Location.

General Information

Base Classifier	Location
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
offset	NumericalValue	1	Specifies the offset / distance of the Location to the anchor of the location.

anchor	AnchorType	1	Specifies if the location on the *TopologySegment* is defined as on offset relative to the startNode of the TopologySegment or the endNode.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologySegment	referencedSegment	1	0..*	N	References the *TopologySegment* on which the Location is located.

7.31.11 Class SegmentMapping

Defines the relationship of an inner segment to its outer topology. The relationship to the outer topology is expressed with a *Path.*

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologySegment	innerSegment	1		N	
Path	mappedPosition	1	0..1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TopologyMapping Specification			0..*	Y	

7.31.12 Class TopologyBendingRestriction

As the name implies, a *TopologyBendingRestriction* defines restrictions on the bendability a path in the *Topology.* There are multiple reasons, why such restrictions exist, for example:

- There is a bending restriction for a wire in the segment.
- The number of wires and the segment diameter is such, that excessive bending causes intolerable torsion forces on wires in the segment.
- Other technical reasons

Since these restrictions can be determined using a variety of methods the VEC provides a concept to store this information for later use. This makes the information available in the downstream processes, without detailed knowledge of the determination procedure (e.g. during form board design, packaging or installation).

The restriction applies to a path of segments as this can cover different case:

- a single segment
- two adjacent segments at a node

- the complete path of a wire

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
minBendRadiusDynamic	NumericalValue	0..1	Specifies the minimum bend radius for the restricted path, if it is used in a dynamic environment, where it is bended repeatedly (e.g. in the grommet of the back door).
minBendRadiusStatic	NumericalValue	0..1	Specifies the minimum bend radius for the restricted path, if it is used in a static environment, where it is bended once during installation. After that it remains unchanged in its bended position during usage.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Path	restrictedPath	1	0..1	Y	The path that defines the restricted way in the topology.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TopologyBendingRestrictionSpecification	1		0..*	Y	

7.31.13 Class TopologyBendingRestrictionSpecification

A *TopologyBendingRestrictionSpecification* can be used to define *TopologyBendingRestrictions.*

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologyBending Restriction		0..*	1	Y	

7.31.14 Class TopologyGroupSpecification

A TopologyGroupSpecification defines a new Topology based on the grouped Topologies. (see KBLFRM-240)

General Information

Base Classifier	TopologySpecification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologySpecification	topologySpecification	0..*	0..*	N	

7.31.15 Class TopologyMappingSpecification

A *TopologyMappingSpecification* allows the definition of hierarchical topologies. It relates an outer topology with an enclosed inner topology.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
NodeMapping		0..*	1	Y	
TopologySpecification	outerTopology	1		N	
TopologySpecification	innerTopolgy	1		N	
SegmentMapping		0..*		Y	

7.31.16 Class TopologyNode

A TopologyNode is a point in the Topology where TopologySegments are starting and ending.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
aliasId	AliasIdentification	0..*	Specifies additional identifiers for the TopologyNode. Example: TopologyNode Ids may vary from one CAD system export to another. Therefore, the CAD system Id is improper for

			identification attribute. The identification shall have a value which is unique within the Topology. AliasId may be used for the CAD system Id.
identification	String	1	Specifies a unique identification of the TopologyNode. The identification is guaranteed to be unique within the TopologySpecification. For all VEC-documents a TopologyNode-instance can be trusted to be the same if the TopologySpecification-instance is the same (see TopologySpecification) and the identification of the TopologyNode is the same.
matchingPointId	String	0..1	Specifies an identification of a TopologyNode which be used for matching nodes that belong to different TopologySpecifications and that are representing the same node. Example: There are two TopologySpecifications, each specifying the topology of a certain zone of the car. If the zones are adjacent, it is possible that there are TopologyNodes where the two topologies are connected. These "connection-nodes" would carry the same matchingPointId.
processingInstruction	LocalizedString	0..*	Specifies processing instructions for the TopologyNode.
nodeType	NodeType	0..1	Specifies the type of the TopologyNode. A Node can either be an EndNode, a Junction or an Inliner.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
UsageNode	realizedUsageNode	0..1	0..*	N	References the *UsageNode* that is realized by this *TopologyNode*.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
NodeMapping		innerNode	1	N	
NodeLocation	0..*	referencedNode	1	N	References the TopologyNode on which the Location is located.
TopologySegment	0..*	startNode	1	N	References the TopologyNode where the TopologySegment starts.
GeometryNode	0..*	referenceNode	0..1	N	References the TopologyNode that is represented by the GeometryNode.
TopologyNode	0..*	instantiatedNode	0..1	N	If this *TopologyNode* is an instance of another *TopologyNode* that is defined elsewhere (e.g. the topology of an assembly), then the instantiated may be referenced here.
TopologySegment	0..*	endNode	1	N	References the TopologyNode where the TopologySegment ends.
TopologySpecification	1	topologyNode	0..*	Y	Specifies the TopologyNodes defined by the TopologySpecification.

7.31.17 Class TopologySegment

A *TopologySegment* is a distance in the Topology where no intermediate electrical contacts appear. If a Topology contains routed wires, then the wire at the beginning of a TopologySegment must be the same as in the ending.

TopologySegments are a logical construct to describe the physical representation of a wiring harness topology. Therefore, a *TopologySegment* is only valid if it has a physical manifestation. *TopologySegments* with a length of 0 or less do not have a physical manifestation and are therefore not valid.

Additionally, the usage of *TopologySegments* with a length of 0 create problems in the overall process. For example, the synchronization of 3D / 2D systems becomes harder or even impossible and the handling of wire protections on those *TopologySegment* is also unclear.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the TopologySegment. The identification is guaranteed to be unique within the TopologySpecification. For all VEC-documents a TopologySegment-instance can be trusted to be the same if the TopologySpecification-instance is the same (see TopologySpecification) and the identification of the TopologySegment is the same.
form	SegmentForm	1	Specifies the form of the TopologySegment. A TopologySegment can either be circular or noncircular.
aliasId	AliasIdentification	0..*	Specifies additional identifiers for the TopologySegment. Example: TopologySegment Ids may vary from one CAD system export to another. Therefore, the CAD system Id is improper for identification attribute. The identification shall have a value which is unique within the Topology. AliasId may be used for the CAD system Id.
processingInstruction	LocalizedString	0..*	Specifies processing instructions for the TopologySegment.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologyNode	startNode	1	0..*	N	References the TopologyNode where the TopologySegment starts.
SegmentLength	lengthInformation	0..*	1	Y	Specifies the different SegmentLengths of the TopologySegment.
SegmentCrossSectionArea	crossSectionAreaInformation	0..*	1	Y	Specifies the different SegmentCrossSectionAreas of the TopologySegment.
TopologyNode	endNode	1	0..*	N	References the TopologyNode where the TopologySegment ends.

Incoming Relations

Other End		This End		General		
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**	
TopologySegment		0..*	instantiatedSegment	0..1	N	If this *TopologySegment* is an instance of another *TopologySegment* that is defined

					elsewhere (e.g. the topology of an assembly), then the instantiated may be referenced here.
TopologySpecification	1	topologySegment	0..*	Y	Specifies the TopologySegments defined by the TopologySpecification.
SegmentMapping		innerSegment	1	N	
Routing	0..*	mandatorySegment	0..*	N	Specifies some constraints for the routing. If the path of the routing is recalculated the referenced segments must be visited.
ZoneAssignment	0..*	assignedSegment	1	N	The *TopologySegment* that is assigned to *TopologyZone* with this *ZoneAssignment.*
GeometrySegment	0..*	referenceSegment	0..1	N	References the TopologySegment that is represented by the GeometrySegment.
Path	0..*	segment	0..*	N	Specifies an ordered list of TopologySegments the routing goes through.
SegmentLocation	0..*	referencedSegment	1	N	References the *TopologySegment* on which the Location is located.

7.31.18 Class TopologySpecification

Specification for the definition of a topology. A topology consists of TopologyNodes, TopologySegments.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologySegment	topologySegment	0..*	1	Y	Specifies the TopologySegments defined by the TopologySpecification.
TopologyNode	topologyNode	0..*	1	Y	Specifies the TopologyNodes defined by the TopologySpecification.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TopologyMapping Specification		outerTopology	1	N	
TopologyMapping Specification		innerTopolgy	1	N	
TopologyGroupSpecification	0..*	topologySpecification	0..*	N	

7.31.19 Class TopologyZone

A *TopologyZone* divides a topology in different sections / design spaces. A *TopologyZone* may be subdivided in further *TopologyZone*. There can be multiple reasons for creating an orthogonal sectioning on a *Topology*. Therefore, *TopologyZones* can be overlapping.

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the Zone (normally the name). The identification is guaranteed to be unique within the TopologySpecification. For all VEC-documents a Zone-instance can be trusted to be the same if the TopologySpecification-instance is the same (see TopologySpecification) and the identification of the Zone is the same.
type	TopologyZoneType	0..1	The type of the TopologyZone. Valid values are defined in an OpenEnumeration.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the zone.
ambientTemperature	TemperatureInformation	0..1	Defines the ambient temperature that can occur in this zone. This can result in specific requirements for the used components.
requiredRobustnessProperties	RobustnessProperties	0..*	Defines the robustness properties that are required in this zone. This can result in specific requirements for the used components (e.g. the ability for sealing).

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ZoneAssignment	assignment	0..*		Y	The assignments of specific topology elements to this zone.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
TopologyZoneSpecification	0..1	zone	0..*	Y	Specifies the *TopologyZones* that are part of the *TopologyZoneSpecification*.
TopologyZone	0..1	subZone	0..*	Y	Specifies the sub *TopologyZones* that are part the *TopologyZone*. All *ZoneAssignments* defined for subZones are automatically inherited by the parent zone.
BuildingBlockSpecification3D	0..*		0..1	N	References the Zone that is building block represents. This shall be a TopologyZone with the type "DmuZone".

7.31.20 Class TopologyZoneSpecification

Specification for the definition of TopologyZones.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologyZone	zone	0..*	0..1	Y	Specifies the *TopologyZones* that are part of the *TopologyZoneSpecification*.

7.31.21 Class ZoneAssignment

A *ZoneAssignment* defines that a specific TopologySegment is affected by the *TopologyZone.* If a coverage is defined, only the parts of the Segment that have a coverage are affected. If no coverage is defined, the whole segment is affected.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
TopologySegment	assignedSegment	1	0..*	N	The *TopologySegment* that is assigned to *TopologyZone* with this *ZoneAssignment.*
ZoneCoverage	coverage	0..*	1	Y	Contains a set of *ZoneCoverages* that define the areas of a *TopologySegment* that is affected by the TopologyZone. If no coverage is defined, the complete segment is affected. Multiple coverages can be necessary if the *TopologySegment* zigzags in and out of the *TopologyZone.*

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ZoneAssignment				Y	
TopologyZone		assignment	0..*	Y	The assignments of specific topology elements to this zone.

7.31.22 Class ZoneCoverage

A *ZoneCoverage* defines an area on a *TopologySegment* that lies within a *TopologyZone.* The area is defined with two *Locations*. *Locations* are the same mechanism that is used to define placements for components. The area that is in the *TopologyZone* is the area between the two locations. There is no semantic in the direction of the definition, so the assignment of first & second Location is completely arbitrary.

However, there are some restrictions for the definition of the locations. All locations have to be in relation to the *TopologySegment* that is referenced by the containing *ZoneAssignment*. This means the *Locations* have to be

either a *SegmentLocation on the* respective *TopologySegment* or a *NodeLocation* on the start or end node of this particular *TopologySegment.* A *ZoneCoverage* from start to end node of a *TopologySegment* is equivalent to the complete omission of *ZoneCoverages* for a particular *ZoneAssignment*.

General Information

Base Classifier	
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Location	secondLocation	1	0..1	Y	
Location	firstLocation	1	0..1	Y	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ZoneAssignment	1	coverage	0..*	Y	Contains a set of *ZoneCoverages* that define the areas of a *TopologySegment* that is affected by the TopologyZone. If no coverage is defined, the complete segment is affected. Multiple coverages can be necessary if the *TopologySegment* zigzags in and out of the *TopologyZone.*

7.31.23 Enumeration AnchorType

Enumeration for the definition of AnchorType of the SegmentLocation.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
FromStartNode		The offset of the location is measured from the startNode of the TopologySegment.
FromEndNode		The offset of the location is measured from the endNode of the TopologySegment.

7.31.24 Enumeration LengthClassification

Enumeration for the definition of a LengthClassification.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**

Designed		A designed length means that the value is derived automatically in a CAD Tool (e.g. from a 3D Geometry).
Adapted		An adapted length means that the value is not the exact value taken from the CAD tool but is adapted in some way. Adapted values are supposed for further use in the process, especially as basis for the product specification. Adapted values are normally created for example by rounding the designed values.

7.31.25 Enumeration NodeType

Enumeration for the definition of the type of a TopologyNode.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
EndNode		Electrical components are normally placed on an EndNode.
Junction		A Junction is a TopologyNode where no electrical component is placed.
Inliner		An Inliner is a TopologyNode where one section of the electrical system is connected to another section.

7.31.26 Enumeration SegmentCrossSectionAreaType

Defines valid values the type of the cross-section area of a *TopologySegment*, since a *TopologySegment* can have different cross section areas with different meanings in the process.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Reserved		*Reserved* is the type for cross section areas that define a reserved space in the DMU for the *TopologySegment.*
Real		*Real* is the type for cross section areas that can be observed for *TopologySegments* with variants of a Harness that are producible.

7.31.27 Enumeration SegmentForm

Enumeration for the definition of the SegmentForm.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
Circular		

NonCircular		

7.31.28 Enumeration TopologyZoneType

There can be various reasons to define TopologyZones. These can be differentiated with these type literals.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
DmuZone		
EnvironmentZone		

7.32 Module usage_constraint

7.32.1 Class UsageConstraint

Specifies a constraint of the possible usages for PartVersion or PartUsages. UsageConstraints are of part master data information. UsageConstraints specify general restrictions for the application of a PartVersion or PartUsage (e.g. a certain period of time in which the part is allowed to be used). (see KBLFRM-229)

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	**Type**	**Mult**	**Comment**
type	UsageConstraintType	1	Defines if the usage constraint is positive (allowance) or negative (denial).
fromDate	Date	0..1	Specifies the lower bound of the time period to which the usage constraint applies.
toDate	Date	0..1	Specifies the upper bound of the time period to which the usage constraint applies.
fromSerialNumber	String	0..1	Specifies the lower bound of a serial number range to which the usage constraint applies.
toSerialNumber	String	0..1	Specifies the upper bound of a serial number range to which the usage constraint applies.
projectPhase	String	0..*	Specifies the project phases to which the usage constraint applies.

Outgoing Relations

Other End			**This**	**General**	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**

UsageNode	usageNode	0..*	0..*	N	References the *UsageNode* to which the *UsageConstraint* applies. This means the described *PartVersion* is allowed / denied in the referenced UsageNode.
Project	project	0..*	0..*	N	References the *Projects* to which the *UsageConstraint* applies. This means the described PartVersion is allowed / denied in the referenced UsageConstraint.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
UsageConstraintSpecification	1	partUsageConstraint	1..*	Y	Specifies the UsageConstraints that apply to the PartVersion or PartUsages described by the UsageConstraintSpecification. The ordering of this association is relevant. The elements shall arranged in the order of ascending priority. That means, elements further back in the collection have a higher priority.

7.32.2 Class UsageConstraintSpecification

Specification for the definition of usage constraints. The associated UsageConstraints are restricting the possible usages of the associated PartVersions and PartUsages.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
PartVersion	constrainedParts	0..*	0..*	N	References the *PartVersions* to which this *UsageConstraintSpecification* applies.
UsageConstraint	partUsageConstraint	1..*	1	Y	Specifies the UsageConstraints that apply to the PartVersion or PartUsages described by the UsageConstraintSpecification. The ordering of this association is relevant. The elements shall arranged in the order of ascending priority. That means, elements further back in the collection have a higher priority.

7.32.3 Enumeration UsageConstraintType

Enumeration for the definition of the type of a UsageConstraint. Valid values are allow and deny.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		**Comment**
Allow		

Deny		

7.33 Module usage_node

7.33.1 Class UsageNode

A *UsageNode* represents a position in an abstract vehicle. For example, the "Head Light Left". *UsageNodes* belong to the master data and they are defined on some companywide level. They can be used to enforce consistent naming over different projects and different development streams (e.g. between Geometry and Electrologic).

A *UsageNode* can be realized by different elements in the VEC (e.g. *NetworkNode, OccurrenceOrUsage, TopologyNode, ComponentNode*).

General Information

Base Classifier	ConfigurableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
abbreviation	LocalizedString	0..*	Specifies an abbreviation of the *UsageNode*. Normally this a human readable short name.
identification	String	1	Specifies a unique *identification* of the *UsageNode*. The *identification* is guaranteed to be unique within the context. For all VEC-documents a *UsageNode-instance* can be trusted to be the same if the context-instance is the same and the *identification* of the *UsageNode* is the same.
description	AbstractLocalizedString	0..*	Specifies additional, human readable information about the *UsageNode*.
usageNodeType	UsageNodeType	0..1	Defines the type of the UsageNode. The type determines how the UsageNode is handled in the latter processes.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Project	usedInProject	0..*	0..*	N	Specifies the *Projects* in which the *UsageNode* can be used.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
UsageNode	0..1	subUsageNodes	0..*	Y	
NetworkNode	0..*	realizedUsageNode	0..1	N	References the *UsageNode* that is realized by this *NetworkNode*.
OccurrenceOrUsage	0..*	realizedUsageNode	0..1	N	References the *UsageNode* that is realized by this *OccurrenceOrUsage*.
UsageConstraint	0..*	usageNode	0..*	N	References the *UsageNode* to which the *UsageConstraint* applies. This means the described

					PartVersion is allowed / denied in the referenced UsageNode.
ComponentNode	0..*	realizedUsageNode	0..1	N	References the *UsageNode* that is realized by this *ComponentNode*.
UsageNodeSpecification	0..1	usageNodes	0..*	Y	Specifies the *UsageNodes* defined by this *UsageNodeSpecification.*
TopologyNode	0..*	realizedUsageNode	0..1	N	References the *UsageNode* that is realized by this *TopologyNode*.

7.33.2 Class UsageNodeSpecification

A *UsageNodeSpecification* is a container for the specification of *UsageNodes.*

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
UsageNode	usageNodes	0..*	0..1	Y	Specifies the *UsageNodes* defined by this *UsageNodeSpecification.*

7.33.3 Enumeration UsageNodeType

Enumeration for the different types of *UsageNodes.*

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		**Comment**
ECU		
Sensor		
Actuator		
CouplingDevice		
EnergyStorage		
Generator		
PowerDistribution		
Switch		
Lamp		

Relay		
Fuse		
Ground		Defines that this ComponentNode is a grounding point.
Splice		
Protection		
Fixing		
Grommet		
CableDuct		

7.34 Module variants

7.34.1 Class ApplicationConstraint

An *ApplicationConstraint* defines a condition with which it is possible to express the inclusion or exclusion of ConfigurableElements in different variants of a product. The *ApplicationConstraint* is focused to express validity rules based on time periods or elements of the product hierarchy in a concise way (attributes and relationships). It is complementary to the *VariantConfiguration* which expresses validity rules based on control strings.

An *ApplicationConstraint* can reference another *ApplicationConstraint* as *baseInclusion*. In this case, an *ApplicationConstraint* can only be satisfied if its *baseInclusion* is also satisfied.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
type	ApplicationConstraintType	1	Defines if the application constraint is positive (allowance) or negative (denial). If the *ApplicationConstraint* has a *baseInclusion* that *baseInclusion* shall define the same type.
fromDate	Date	0..1	Specifies the lower bound of the time period to which the application constraint applies.
toDate	Date	0..1	Specifies the upper bound of the time period to which the application constraint applies.
fromSerialNumber	String	0..1	Specifies the lower bound of a serial number range to which the application constraint applies.
toSerialNumber	String	0..1	Specifies the upper bound of a serial number range to which the application constraint applies.
projectPhase	String	0..*	Specifies the project phases to which the application constraint applies.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
Project	project	0..*		N	Defines the projects for which the application constraint applies.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConfigurableElement		applicationConstraint	0..*	N	References the application constraints that apply to the ConfigurableElement.
ApplicationConstraint	0..*	baseInclusion	0..1	N	An *ApplicationConstraint* can only be satisfied if its *baseInclusion* is satisfied as well.
ApplicationConstraintSpecification	1	applicationConstraint	1..*	Y	Specifies the UsageConstraints that apply to the PartVersion or PartUsages described by the UsageConstraintSpecification.

7.34.2 Class ApplicationConstraintSpecification

Specification for the definition of application constraints. The associated ApplicationConstraints are restricting the possible usages of the associated *ConfigurableElements*.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
ApplicationConstraint	applicationConstraint	1..*	1	Y	Specifies the UsageConstraints that apply to the PartVersion or PartUsages described by the UsageConstraintSpecification.

7.34.3 Class VariantCode

VariantCodes are defining the literals on which VariantConfiguration are stated. Possible VariantCodes might be different for steering types, optional equipment, engine-power class.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the variant code. This is also the literal value for the VariantCode, which shall be used in the logisticControlExpressions of VariantConfigurations.

aliasId	AliasIdentification	0..*	Room to specify additional identifiers for the VariantCode.
abbreviation	LocalizedString	0..1	Room for a human readable short name, title etc. of the VariantConfiguration.
codeType	String	0..1	Allows the classification of a VariantCodes (e.g. Base-Option, Extra-Option, ...).
description	AbstractLocalizedString	0..*	On optional human readable description of the variant code.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
VariantGroup	0..*	variantCode	0..*	N	References the VariantCodes that are member of the VariantGroup.
VariantCodeSpecification	1	variantCode	0..*	Y	Specifies the individual VariantCodes defined in the VariantCodeSpecification.

7.34.4 Class VariantCodeSpecification

Specification for the definition of variant codes.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
VariantCode	variantCode	0..*	1	Y	Specifies the individual VariantCodes defined in the VariantCodeSpecification.

7.34.5 Class VariantConfiguration

A variant configuration defines a condition with which it is possible to express the inclusion or exclusion of ConfigurableElements in different variants of a product.

A *VariantConfiguration* can reference another *VariantConfiguration* as *baseInclusion*. In this case, a *VariantConfiguration* can only be satisfied if its *baseInclusion* is also satisfied.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment

identification	String	0..1	Specifies a unique identification of the variant configuration. The identification is guaranteed to be unique within the specification and does not change over the time.
description	AbstractLocalizedString	0..*	On optional human readable description of the variant configuration.
logisticControlString	String	0..1	Specifies a logisticControlString which can be used if the variant management is not done by boolean logic.
logisticControlExpression	String	0..1	Specifies a logisticControlExpression expressed as boolean term.
configurationType	String	0..1	Allows the classification of a VariantConfiguration. (see KBLFRM-250, KBLFRM-314, KBLFRM-290)

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
ConfigurableElement	0..*	configInfo	0..1	N	References the configuration information that applies to the ConfigurableElement.
VariantConfiguration		baseInclusion	0..1	N	A *VariantConfiguration* can only be satisfied if its *baseInclusion* is satisfied as well.
VariantConfigurationSpecification	1	variantConfiguration	0..*	Y	Specifies the individual VariantConfigurations defined in the VariantConfigurationSpecification.

7.34.6 Class VariantConfigurationSpecification

Specification for the definition of variant configurations.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
VariantConfiguration	variantConfiguration	0..*	1	Y	Specifies the individual VariantConfigurations defined in the VariantConfigurationSpecification.

7.34.7 Class VariantGroup

With a VariantGroup it is possible to group VariantCodes. The semantic of this grouping should be defined with the groupType (e.g. composition, choice, etc.).

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the variant code.
aliasId	AliasIdentification	0..*	Room to specify additional identifiers for the VariantGroup.
abbreviation	LocalizedString	0..1	Room for a human readable short name, title etc. of the VariantGroup.
description	AbstractLocalizedString	0..*	On optional human readable description of the variant group.
groupType	VariantGroupType	0..1	Allows the classification of a VariantGroups into different types. For example: - composition (e.g. winter package) - choice (e.g. right hand / left hand driving). Agreed literals for this attribute are defined in the OpenEnumeration *VariantGroupType.*

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
VariantCode	variantCode	0..*	0..*	N	References the VariantCodes that are member of the VariantGroup.

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
VariantGroupSpecification	1	variantGroup	0..*	Y	Specifies the individual VariantGroups defined in the VariantGroupSpecification.
VariantStructureNode	0..*	containedGroups	0..*	N	

7.34.8 Class VariantGroupSpecification

Specification for the definition of variant groups.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
VariantGroup	variantGroup	0..*	1	Y	Specifies the individual VariantGroups defined in the VariantGroupSpecification.

7.34.9 Class VariantStructureNode

VariantStructureNodes can be used to define a hierarchical structure on *VariantGroups*. Every *VariantStructureNodes* can reference *VariantGroups* and *VariantStructureNodes* as children.

General Information

Base Classifier	ExtendableElement
Applied Stereotype	
Is Abstract	false

Attributes

Name	Type	Mult	Comment
identification	String	1	Specifies a unique identification of the *VariantStructureNode*.
aliasId	AliasIdentification	0..*	Room to specify additional identifiers for the *VariantStructureNode*.
abbreviation	LocalizedString	0..1	Room for a human readable short name, title etc. of the *VariantStructureNode*.
description	AbstractLocalizedString	0..*	On optional human readable description of the *VariantStructureNode*.

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
VariantGroup	containedGroups	0..*	0..*	N	

Incoming Relations

Other End		This End		General	
Type	**Mult**	**Role**	**Mult**	**Agg**	**Comment**
VariantStructureNode		childNodes	0..*	Y	
VariantStructureSpecification	0..1	rootNode	1	Y	

7.34.10 Class VariantStructureSpecification

Specification to define any hierarchical structure on variant groups (e.g. by the means of a functional organization). The hierarchy starts with a single root node.

General Information

Base Classifier	Specification
Applied Stereotype	
Is Abstract	false

Outgoing Relations

Other End			This	General	
Type	**Role**	**Mult**	**Mult**	**Agg**	**Comment**
VariantStructureNode	rootNode	1	0..1	Y	

7.34.11 Enumeration ApplicationConstraintType

Enumeration for the definition of the type of an ApplicationConstraints. Valid values are allow and deny.

General Information

Applied Stereotype	ClosedEnumeration

Enumeration Literals

Name		Comment
Allow		
Deny		

7.34.12 Enumeration VariantGroupType

Defines valid values for the type of VariantGroups.

General Information

Applied Stereotype	OpenEnumeration

Enumeration Literals

Name		Comment
Family		A group with the type "Family" defines a set of variant codes that are mutually exclusive, and a valid configuration has to choose one variant code from the group.

7.35 Module VEC

7.35.1 Primitive Boolean

Primitive type for attributes with boolean values. During schema generation this is translated to *xs:boolean*.

General Information

Applied Stereotype	

7.35.2 Primitive Date

Primitive type for attributes with date and time values. During schema generation this is translated to *xs:dateTime*.

General Information

Applied Stereotype	

7.35.3 Primitive Double

Primitive type for attributes with double values. During schema generation this is translated to *xs:double*.

General Information

Applied Stereotype	

7.35.4 Primitive Integer

Primitive type for attributes with integer values. During schema generation this is translated to *xs:integer*.

General Information

Applied Stereotype	

7.35.5 Primitive String

Primitive type for attributes with string values. During schema generation this is translated to *xs:string.*

General Information

Applied Stereotype	

prostep IVIP

VDA | Verband der Automobilindustrie

A joint Publication of prostep ivip and Verband der Automobilindustrie (VDA)

prostep ivip association

Dolivostraße 11
64293 Darmstadt
Germany

Phone +49-6151-9287336
Fax +49-6151-9287326
psev@prostep.com
www.prostep.org

ISBN 978-3-9820795-7-8
Version 1.2
PSI 31