PDTnet
Usage Guide

Version: *1.0*
Status: ***Final***
Resp. Author: *Dr. Andreas Schreiber, PROSTEP*
Date: *June 2003*
Distribution: *Public*

History

Version	Status	Authors	Changes
0.1	Draft	A. Schreiber, PROSTEP	Creation
0.2	Draft	A. Schreiber, PROSTEP	Extension
1.0	Final	A. Schreiber, PROSTEP	Extension

Content

1 Scope

PDTnet results are mainly described in three documents:

- PDTnet White Paper
  The White Paper focuses on the results and experiences gained in the PDTnet based on application Scenarios.

- PDTnet Implementation Guide
  The Implementation Guide describes the implementation concepts as well as the PDTnet Entities in detail. This document can be used by software engineers for their implementation.

- PDTnet Usage Guide
  The Usage Guide describes PDTnet objects and their application on a level that enables the mapping of PDM data to the PDTnet Schema. It can be used by software engineers as well as by mechanical engineers.

Various information sources are a prerequisite for understanding the document. To this information belongs

- XML technology, XML Schema
- AP214, Express
- Derivation of the PDTnet Schema from AP214
- Application of AP214 constructs
- Resulting PDTnet constructs

The following chapter will give an overview about these mentioned concepts.

2 General Information

2.1 XML Entity Description

XML elements are described by their attributes and element content. XML Schema additionally allows the derivation of types. To describe the XML elements used by the PDTnet approach the following table is used:

Element name			
Parent	<name of the parent element>		
Base Type	<name of the type from which the element is derived>		
derived by	<derivation type>		
XML Elements	**Type**	**MinOccurs/ maxOccurs**	**Description**
<inherited attribute name>	*Attribute type*	*1/1*	*Description of attribute*
<attribute name>	Attribute type	0/1	Description of attribute
<inherited element name>	*Element Type*	*Unbounded*	*Description of element*
<element>	Element Type	Unbounded	Description of element

Table 1: XML element description.

In this table you have the name of the XML element type as header, if it's an abstract type a **(abstract)** follows the name, and its parent in the PDTnet tree noted as **Parent**. If the element is a derived type, two lines follow with the name of the base type and the way of derivation (restriction or extension). Now the description of basic attributes with their names and types and their use (required or optional) follows. If the specific attribute is inherited from the supertype, it is written in italic. After the attributes, the content is listed, with element types, minimum number of occurrences and maximum number of occurrences. Here too, inherited elements are marked italic. Due to the SOAP binding the Express attributes are implemented as XML elements; required attributes have a minOccurs of 1, optional attributes have a minOccurs of 0. As they were attributes in the original data model, the maxOccurs for them is 1 or "unbounded" for aggregates.

2.2 EXPRESS-G Instantiation Diagrams

The application protocol AP214 is defined using the ISO-Language Express. The data models can be visualized using the graphical representation of Express that is called Express-G.
The following Table is introducing some language constructs in order to enable the reader the interpretation of the following class diagrams.

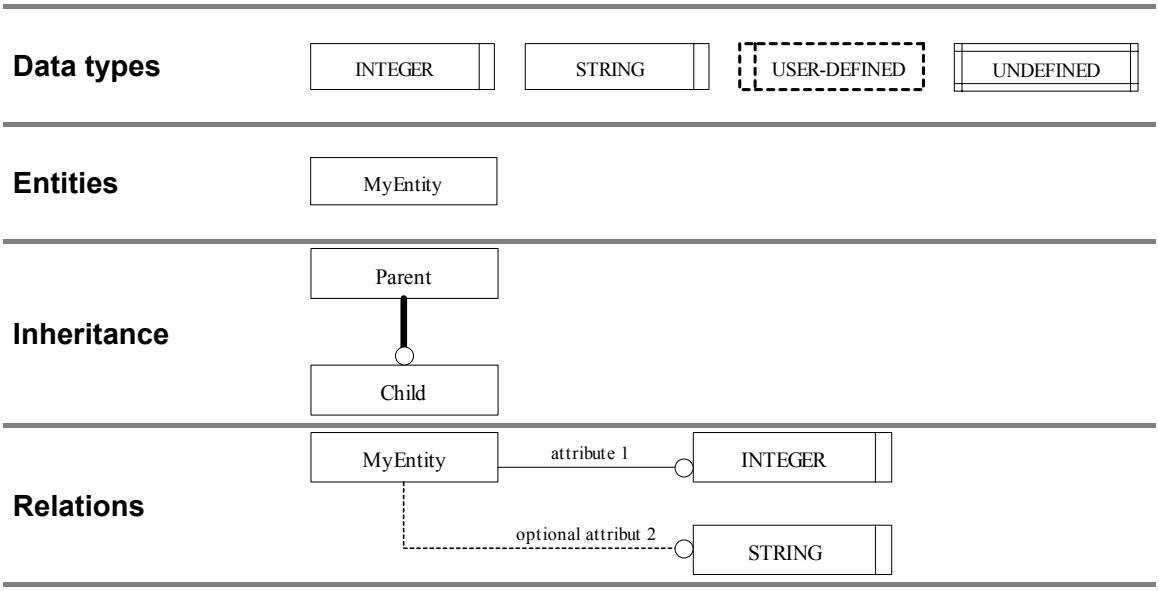


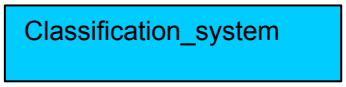
Figure 1: Express-G Syntax.

2.3 XML Instantiation Diagrams

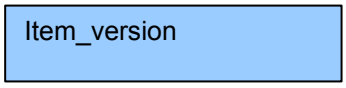
The instantiation diagrams are presented using a graphical notation intended to illustrate the instance model. This notation is not EXPRESS-G. Attributes having a type different from IDREF or IDREFS are not shown in the diagrams.

A legend for the diagram notation is shown below :

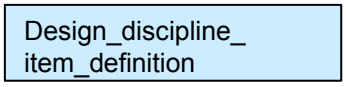
- XML Elements that are direct children of the Pdtnet_schema (first level elements)



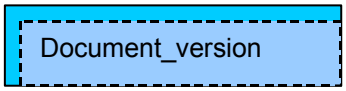
- XML Elements that are direct children of first level elements (second level elements)



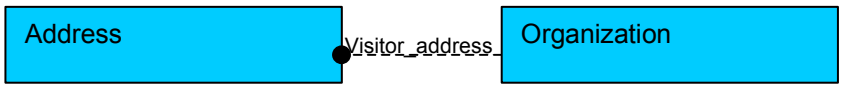
- XML Elements that are direct children of second level elements (third level elements)



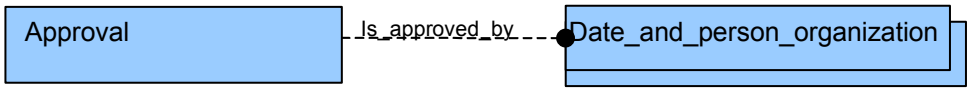
- Parent child relationships



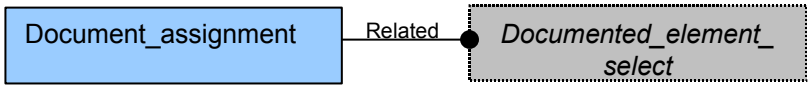
- Relationships as ID/IDREF Relationships (where the name on the arrow gives the name of the referencing attribute; dotted if the reference is optional)



- Relationships as ID/IDREFS Relationships (where the name on the arrow gives the name of the referencing attribute; dotted if the reference is optional)



- References that can point to different XML Elements (a SELECT)



2.4 The Schema

The PDTnet Schema was defined using the W3C XML Schema definition.

The root element of a PDTnet conform XML file must be the Pdtnet_schema element that contains the different PDM data elements. To be able to uniquely identify all elements, each possesses an x-id attribute of type xs:ID.

The following table describes the Pdtnet_schema element and its valid attributes and direct children. A description of the structure of this table that will be used to describe all the used XML elements is given in section (2.1)

PDTnet_schema			
XML Attributes	**Type**	**use**	**Description**
Id	xs:ID	required	Used to identify the XML Element uniquely
Version_id	xs:string	required	Used to identify the schema version
XML Elements			
Activity, Activity_method, Address, Application_context, Approval_status, Axis2_placement_3d, Cartesian_coordinate_space_3d, Cartesian_point, Classification_system, Complex_product, Data_environment, Date_time, Date_time_assignment, Descriptive_specification, Design_constraint, Direction, Document, Document_content_property, Document_creation_property, Digital_file, Document_format_property, Document_location_property, Document_size_property, Document_type_property, Duration, Effectivity, Event_reference, Explicit_transformation_3d, External_library_reference, Implicit_transformation_3d, Item, Language, Material, Organization, Person, Product_class, Product_function, Product_identification, Property, Property_value, Rectangular_size, Unit, Process_plan, Process_plan_version, Process_operation_occurrence, Process_operation_definition, Physical_instance, Product_component, Project, Specific_item_classification, Specification, Specification_category, Specification_expression, Specification_inclusion, Value_with_unit, Work_request			

Table 2: PDTnet Schema structure.

Figure 2 shows an instantiation diagram for the PDTnet object *Person* by using the concept described in chapter 2.4.

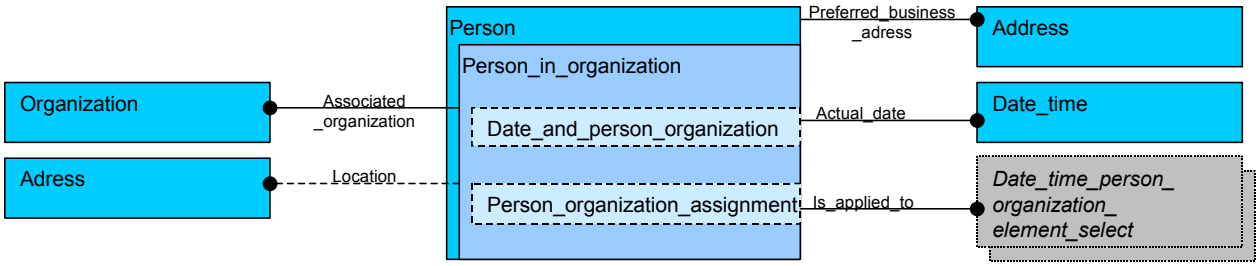


Figure 2: Example PDTnet Instantiation diagram for Person.

The PDTnet Schema is an exact representation of AP214 ARM. For the defined PDTnet Schema Scope (see Table 2) both Schemas define the same objects with the same set of attributes and restrictions for the attribute values, beside one exception.

In order to enable a high performance of standard XML tools by parsing PDTnet XML data an optimization was undertaken. XML is a language working with deep hierarchies while STEP is a reference oriented approach. For this reason it was defined that some 1:n relationships will be transformed to a hierarchy and any n:m relationship will be transformed to hierarchy information with a 1:n link.

Therefore the following containment rule was defined:

The entity definitions of entities with one or more entity-valued attributes may be changed by the following rules:

- *Select one of the entity-valued attributes (no SET or other aggregate).*
- *Remove this element from the element list.*
- *Add the entity to the content model of the referenced type with minOccurs="0" and maxOccurs="unbounded".*
- *If an inverse attribute exists, the element shall be removed from the referenced type and the minOccurs of the contained entity shall be 1 (= default value, so it does not have to be declared).*

Example:

In EXPRESS:

```
ENTITY item;
  id : STRING;
  name : string_select;
  description : OPTIONAL string_select;
  INVERSE
   associated_version : SET[1:?] OF item_version FOR associated_item;
 END_ENTITY;

ENTITY item_version;
  id : STRING;
  associated_item : item;
  description : OPTIONAL STRING;
END_ENTITY;
```

In XML Schema :

```
<xsd:complexType name="Item">
```

```
        <xsd:sequence>
            <xsd:element name="Id" type="xsd:string"/>
            <xsd:element name="Name" type="xsd:string"/>
            <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Item_version" type="Item_version" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:ID" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="Item_version">
        <xsd:sequence>
            <xsd:element name="Id" type="xsd:string"/>
            <xsd:element name="Description" type="String_select" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:ID" use="required"/>
    </xsd:complexType>
```

In XML Instance Document:

```
<Item id="i1">
    <Id>18110028</Id>
    <Name>Airbag Modul Linkslenker</Name>
    <Item_version id="iv11">
        <Id>Version1</Id>
    </Item_version>
</Item>
```

This simple rule needs to be considered by interpreting the instantiation diagrams for AP214 and PDTnet.

3 Using PDTnet Objects

This chapter describes how to use PDTnet objects. Starting from the AP214 object definition the instantiation by using PDTnet will be explained. The following objects are representing only a subset of the PDTnet Schema. But by understanding the concept any STEP consultant is able to use the approach for additional PDTnet entities.

For explaining the EXPRESS-G Diagrams (see chapter 2.2) as well as XML Instantiation Diagrams (see chapter 2.3) are used.

3.1 Part Master Data

Part master identification supports the ability to uniquely identify a part. A part in AP214 is managed as an ***item***. The identification of parts in the STEP PDM Schema consists of three important and structurally distinct concepts:

- Base Identification,
- Version Identification,
- View Definition.

Base Identification
The master base identification (***item***) maintains information common to all part versions and disciplines and/or life-cycle views. It contains the base part number and name. The base number should not be subject to any encoding of information into a single complex parseable string.
The item number identifier must be unique within the scope of the business process of the information exchange. This is typically not a problem when the part data is only used within a single company. If the data is being assembled for an external use, the identification must be interpreted as unique within that broader domain. Processors may need to evaluate more than one string (i.e. item.id) to establish unique identification of a part; there may be a

combination of parameters that make part identification unique. The associated organization entity with the role 'id owner' can be used to derive a uniqueness parameter if the item.id attribute is not unique within the domain of the business arrangement of the exchange.

Version Identification
There must be at least one version (***item_version***) assigned to a part base identification. The version information may represent a design revision or iteration in a design cycle of a part. There may be more than one version associated with a part master. The set of versions associated with a base part may be related together to represent the version history of that product. The product version collects all information among all associated disciplines and life-cycle view definitions.

View Definition
It is recommended that at least one view definition (***design_discipline_item_definition or DDID***) be assigned to each part version. The view collects information relevant from the perspective of a particular application domain or life-cycle stage (e.g. “manufacturing” or “design”). There may be more than one life-cycle view definition associated with a particular version of a part. This is especially important to enable different views on product assembly structures.

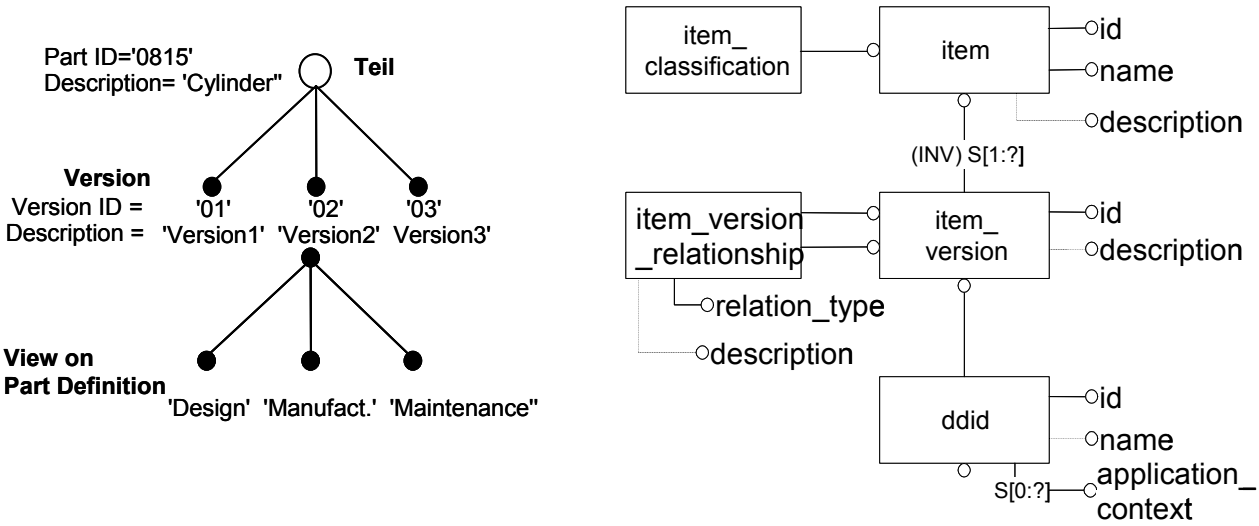


Figure 3: AP214 instantiation of part master data.

While Figure 3 shows an AP214 instantiation example of part master data Figure 4 shows the appropriate PDTnet objects. PDTnet is using exactly the same objects and attributes. The main difference according the concept introduced in chapter 2.4 is related to objects that represent a n:m relationship.
In the case of part master related information it occurs with the elements *document_assignment, alias_identification, item_version_relationship, document_assignment* and *DDID*.

These former independent objects were changed to *item* or *item_version* elements. For this reason the *relating* link of the elements that former *related item* or *item_version* can be deleted from the object definition. The *related* link is used as defined in AP214.

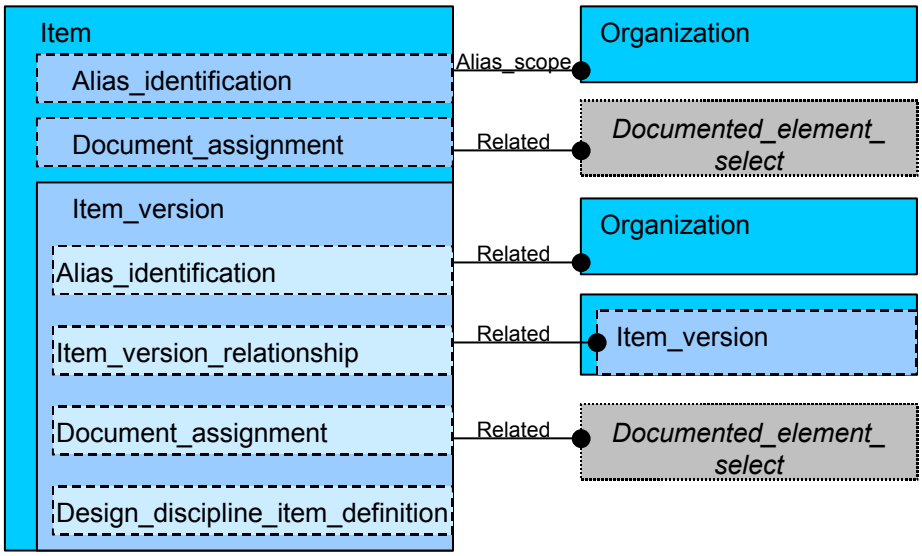


Figure 4: PDTnet instantiation of part master data.

3.2 Part Version Relationships

Using item_version_relationship and document_version_relationship AP214 allows the definition of associations among version information. There are two types of relationships between versions used to represent the version history:

- Sequential relationship: The relating part version is the preceding version and the related part version is the subsequent following version. Both versions must be associated with the same part master base. A part version may have at most one preceding and one subsequent following version.
- Hierarchical relationship: The related part version is a sub version of the relating part version (sometimes referred to as an *iteration*). Both versions must be associated with the same part master base. Each part version may have at most one super version (parent in the hierarchy). A part version may have an arbitrary number of sub versions (children in the hierarchy). Part versions related by hierarchical relationships may not define cyclic relationships.

The attribute item_version_relationship.relation_type describes the relationship more detailed. Where applicable the following values shall be used:

- 'derivation': The application object defines a deriving relationship where the related item_version is based on the relating Item_version which is an earlier version of the same or of a different Item
- 'hierarchy': The application object defines a hierarchical relationship where the related Item_version is a subordinate version of the relating Item_version;

  EXAMPLE 'Rev. 1.1' and 'rev. 1.2' are subordinates of 'version 1'.
- 'sequence': The application object defines a version sequence where the relating Item_version is the preceding version of the related Item_version that is the following version. For a given Item_version there shall be at most one item_version_relationship of this relation_type referring to this Item_version as 'relating' and at most one Item_version_relationship of this relation_type referring as 'related';

- 'supplied item': The application object defines a relationship between two Item_version objects representing the same object in different organizational contexts.

3.3 Part Properties

AP214 allows specifying properties associated with parts. A property is the definition of a special quality and may reflect physics or arbitrary, user defined measurements. A general pattern for instantiating property information is used in AP214. A number of pre-defined property type names are also proposed for use when appropriate.
AP214 allows specifying the ***property_value*** associated with product data via an ***item_property_association***. Product data could be a *ddid* as well as an *item_instance* or others.

General Part Properties{ XE "General Part Properties" }

A ***general_property*** identifies a property type/classification independently of the association of a definition of that type of property to product data.

Pre-defined part properties

AP214 supports the following pre-defined properties:

- **Material_property:**
  The material property specifies from which material the part is made from.
- **Recyclability_property:**
  A recyclability property is information concerning the ability to reuse objects or components of objects after their primarily intended usage.
- **Mass_property:**
  a mass property is a quantity of matter of which an object consists
- **Quality_property:**
  A quality property is a property that provides information about the level of quality of products or processes. A quality property documents, e.g., the level of quality reached for the parts prototypes.
- **Cost_property:**
  a cost property is a property that specifies costs.
- **Duration_property:**
  A duration property is a property that specifies a period of time during which a given object is used or will last. For duration properties property_definition.name **= 'duration property'** shall be used.

All other types can be managed as *general_property*.

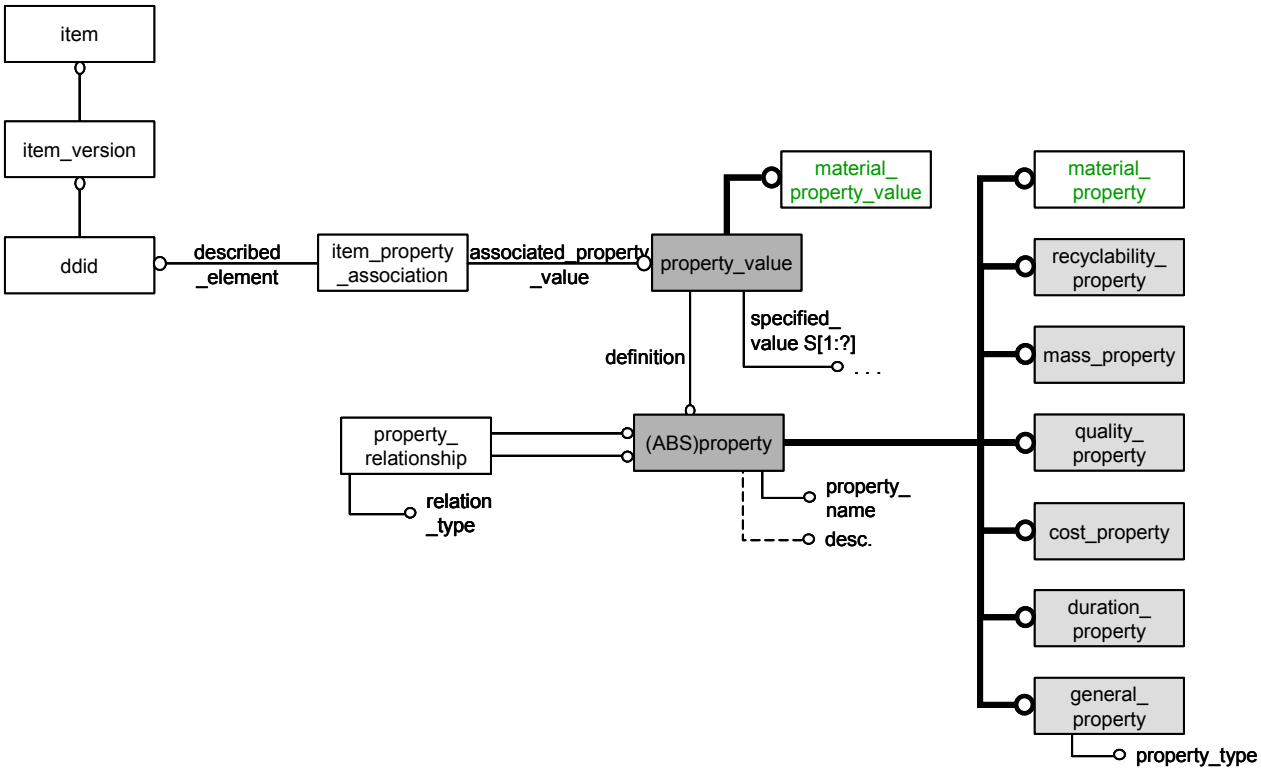


Figure 5: The AP214 property concept.

Regarding the definitions made in chapter 2.4 the element *property_value_association* including its subclasses was changed by deleting the attribute *associated_property_value*. In PDTnet this object is defined as an element of *property_value*.

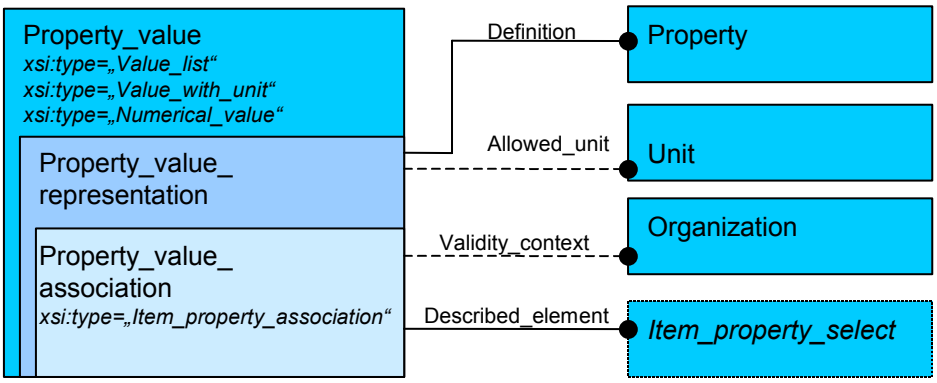


Figure 6: PDTnet instantiation of properties.

3.4 Product Structure

The AP214 supports explicit hierarchical product structures representing assemblies and the constituents of those assemblies. This explicit part structure corresponds to the traditional engineering and manufacturing bill of material indentured parts list. Relationships between part view definitions are the principle elements used to structure an explicit part assembly configuration. The relationship itself represents a specific *usage occurrence* of the constituent part definition within the immediate parent assembly definition. Mechanisms to represent quantity associated with this assembly-component usage relationship are also provided.

AP214 also has the capability to identify and track a specified usage of a component definition in an assembly at a higher level than the immediate parent subassembly. Consider a wheel-axle subassembly composed of one axle and two wheels, the right and the left. A higher-level chassis assembly is in turn composed of two wheel-axle subassemblies, the front and the rear. The requirement to individually identify the left-front wheel, for example, is supported by this capability.

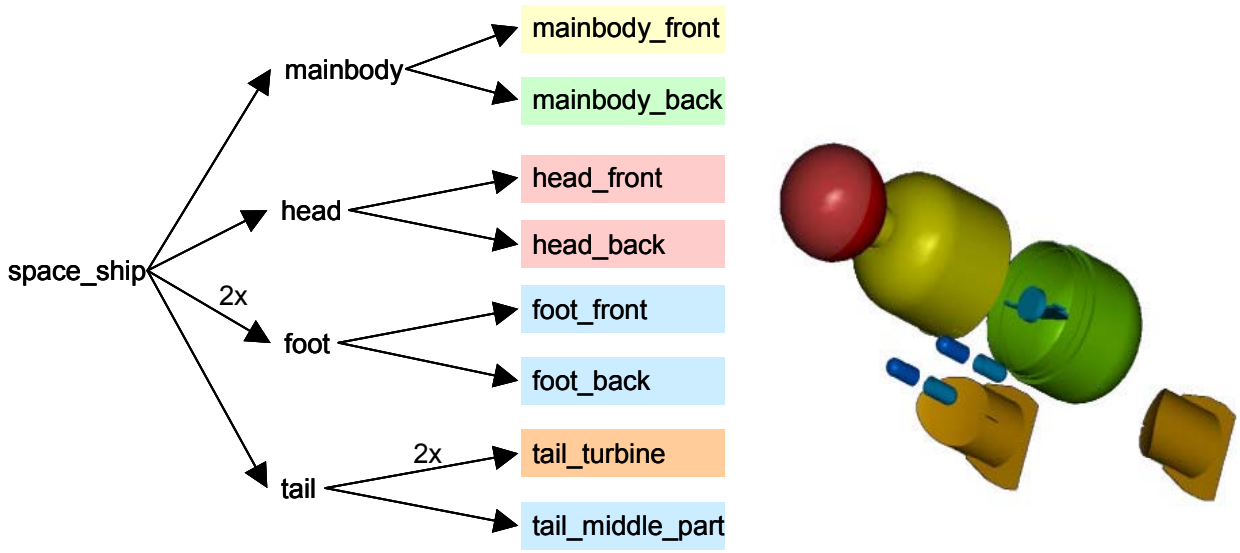


Figure 7: Example for product structure information.

Different view definitions of the same version of a part may participate in different explicit product structures. For example, a design/as-planned view of a particular version of a part, representing the design discipline part definition, may be engaged in an explicit design assembly structure. A manufacturing/as-built view of the same part version represents the definitional template for the actual physical part, and may participate in a manufactured assembly structure that is different from the design assembly structure. Finally, a support/as-maintained view of the part version representing the physical part definition may participate in yet another different disassembly structure.

In addition to hierarchical assembly structures, AP214 supports relationships between parts to characterize explicit alternates and substitutes for the assembly. Other relationships between part definitions exist to characterize the make from relationship and for supplied part identification.

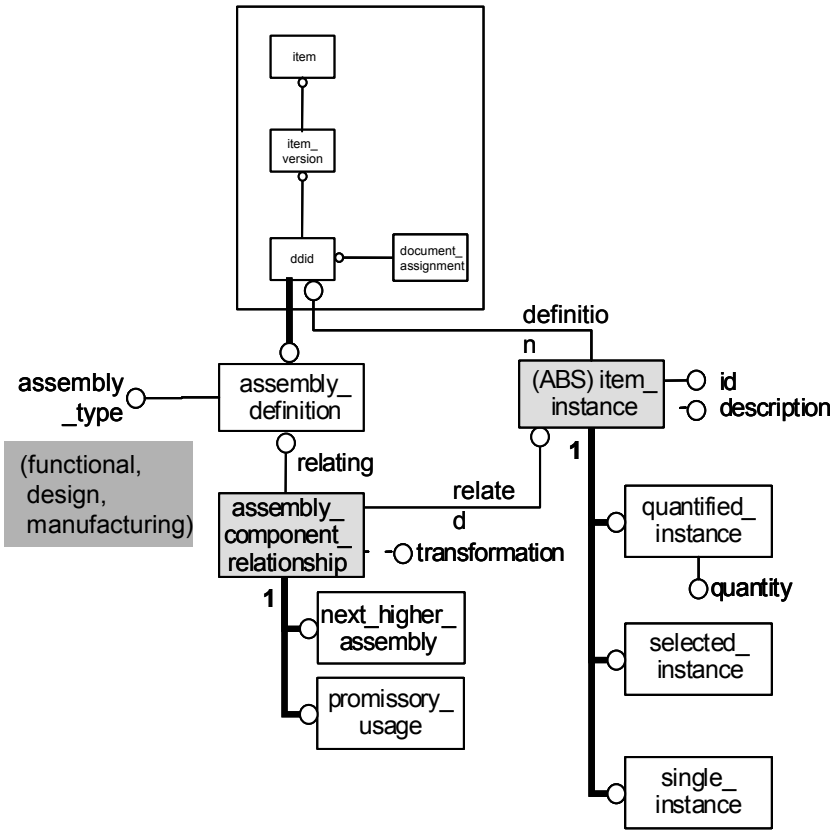


Figure 8: Instantiation of Product Structure information in AP214.

This entity is a subtype of assembly_component_usage. It represents a single individual occurrence of a component definition as used in an immediate next higher parent assembly.

Explicite assembly Bill of Material can be created using the ***next_higher_assembly*** object that refers with next_higher_assembly.relating to the super assembly (*DDID*) and with next_higher_assembly.relating to the subassembly or leaf (***item_instance*** subtypes). Using this construct any structure can be created. The *item_instance.id* attribute contains a unique instance identifier for the individual component occurrence.

The mostly used item_instance subtypes are

- *Single_instance*
- *Quantified_instance*

Single Instance
The single instance object can be used for single parts in an assembly as well as for parts with multiple usage. The AP214 uses separate single usage occurrences to represent multiple occurrences of a component within an assembly when the various components must be distinguished individually. Multiple occurrences of a component within an assembly must be individually identified to allow separate ids, display names, reference designation, or related property information, including shape and orientation/transformation information.

Quantified Instance
The basic explicit assembly represents a single occurrence of the component definition within the assembly definition. Multiple occurrences of a constituent used in an assembly are represented by using the *quantified_instance* construct.

The quantified usage associates a quantity with the component usage, but does not allow independent identification of individual occurrences when a component definition is used multiple times.

Regarding the definitions made in chapter 2.4 the element *item_definition_instance_ relationship* including its subclasses was changed by deleting the attribute *relating*. In PDTnet this object is defined as an element of *DDID* (see Figure 9). The *related* attribute is used as defined in AP214 for referring to child (leafs or subassemblies).

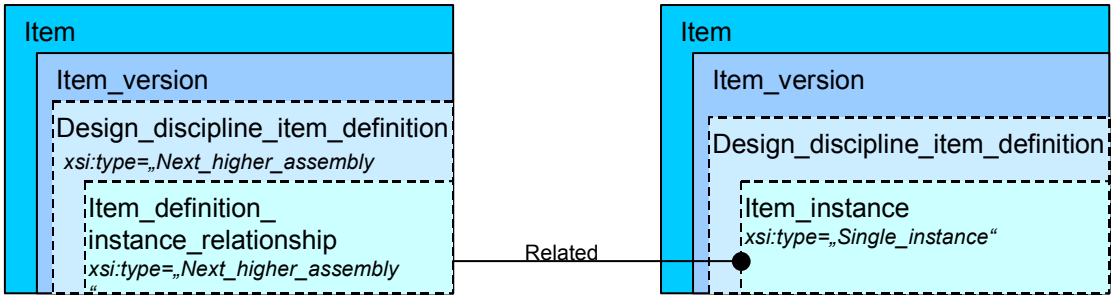


Figure 9: PDTnet Instantiation of Product Structure information.

3.5 Approval

An ***Approval*** is a judgment concerning the quality of those product data that are subject of the Approval. An Approval represents a statement made by technical personnel or management personnel whether certain requirements are met. The absence of approval information does not imply any approval status by default.

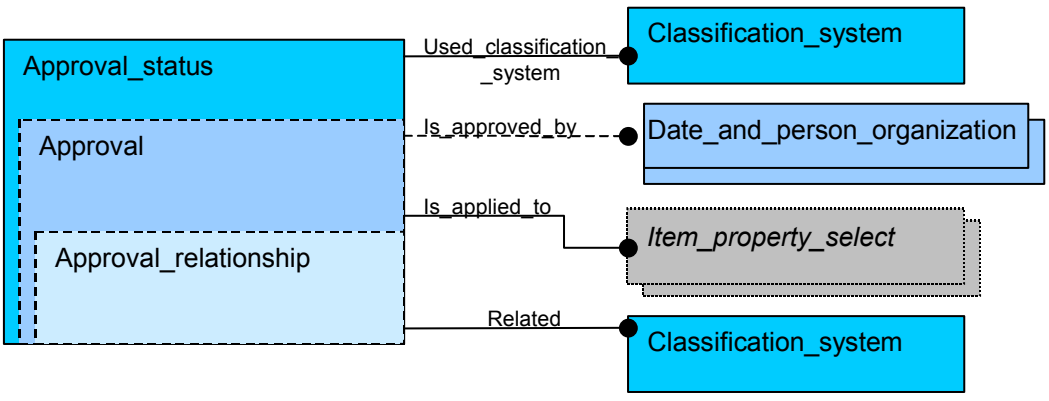


Figure 10: Approval information in PDTnet.

Regarding the definitions made in chapter 2.4 the element *item approval_relationship* including its subclasses was changed by deleting the attribute *relating*. In PDTnet this object is defined as an element of *approval* (see Figure 10). The *related* attribute is used as defined in AP214.

3.6 Date and Time

A ***Date_time*** object is the specification of a date and an optional time of day. By using the ***date_time_assignment*** object it can be assigned to other product data.

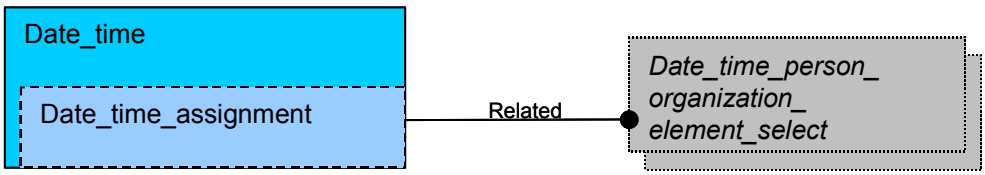


Figure 11: Date and Time in PDTnet.

Regarding the definitions made in chapter 2.4 the element *item date_time_assignment* was changed by deleting the attribute *relating*. In PDTnet this object is defined as an element of *date_time* (see Figure 11). The *related* attribute is used as defined in AP214.

3.7 Organization

An ***Organization*** is a group of people involved in a particular business process.
The data associated with an Organization are the following:

- delivery_address
- id
- organization_name
- organization_type
- postal_address
- visitor_address

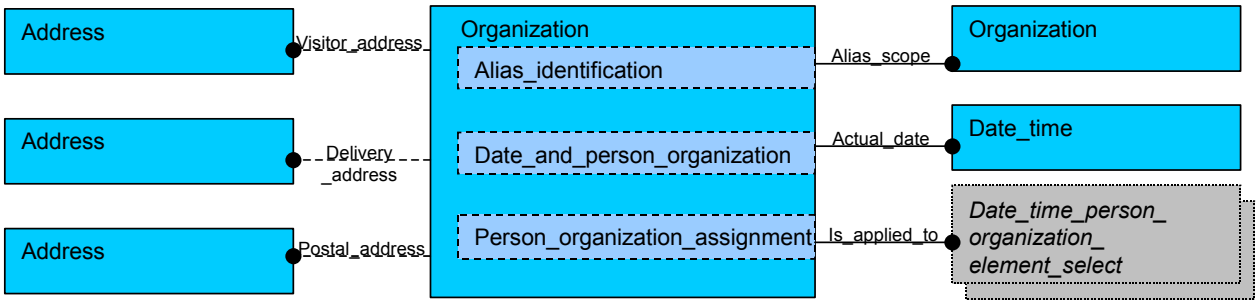


Figure 12: Organization in PDTnet.

Regarding the definitions made in chapter 2.4 the element *person_organization_assignment* was changed by deleting the attribute *assigned_person_organization*. In PDTnet this object is defined as an element of *organization or person* (see Figure 12 and Figure 13). The *related* attribute is used as defined in AP214.

3.8 Person

Person is an individual human being who has some relationship to product data. The Person shall always be identified in the context of one or more organizations.

The data associated with a Person are the following:

- person_name
- preferred_business_address

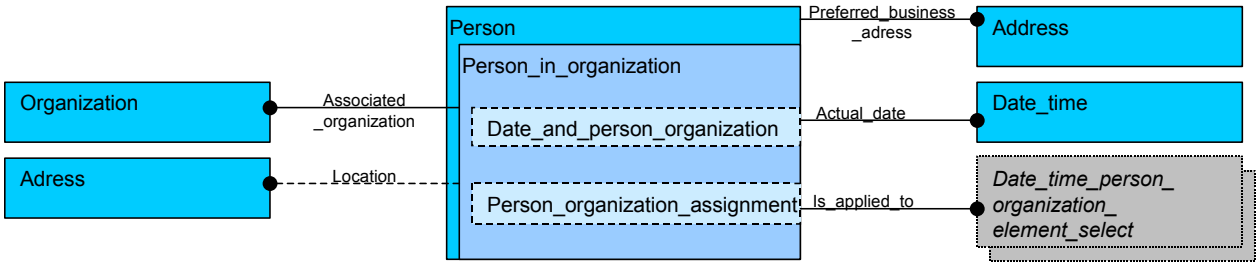


Figure 13: Person in PDTnet.

3.9 Document Master Data

Documents in AP214 are represented using the ***document*** object. As with Parts the concepts of base identification, version identification (***document_version***), and view definition are structurally distinct in the object model. The general recommendations given for part master identification apply to the document master identification, except where differences are noted.

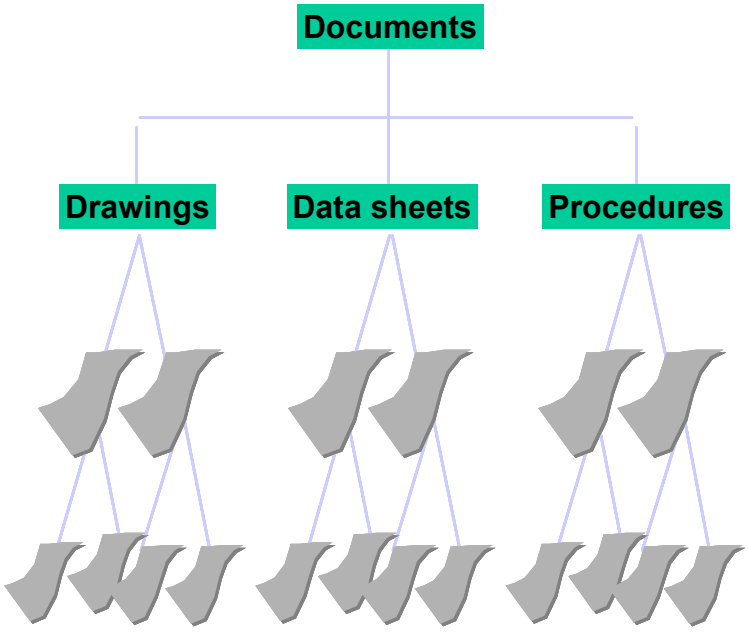


Figure 14: Different types of document representations.

Base document identification is always associated with at least one document version. Multiple document versions of a base document identification may be related together to represent document version history.

There is no view definition as we have with the *ddid* for parts.

Documents may be associated with product data by using the ***document_assignment*** object. Constraints may also be specified on this association, in order to distinguish an applicable portion of an entire document or file in the association.

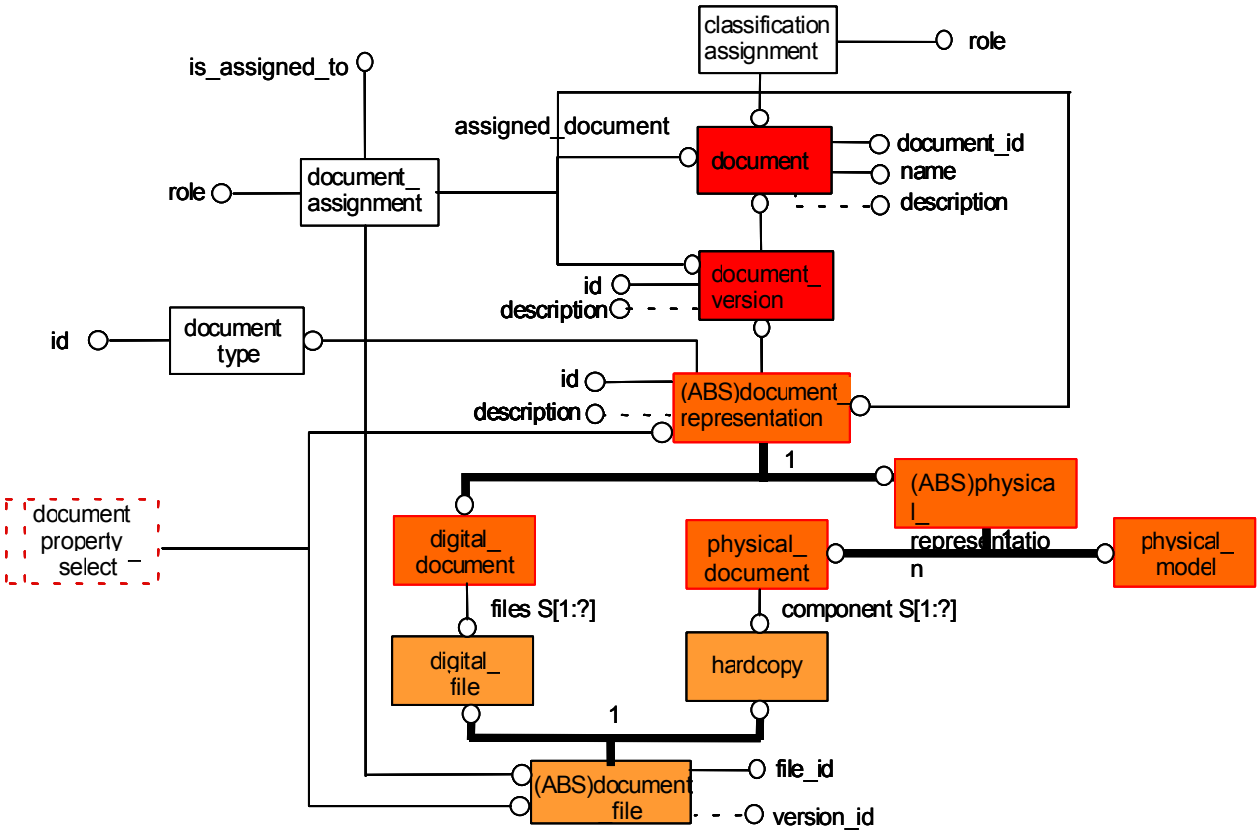


Figure 15: Document Management in AP214.

The following table describes the entities used in the document concept.

AP214 Object	Description
document	Document master data, identified by the document id and described by document name and description
document_version	Versioning information of the document
document_ representation	Representation type of the document_version, can instantiate the subtypes digital_document, physical_document and physical_model
.digital_document	Representation as digital document that can consist on one ore more than one files (digital files).

.physical_document	Representation als paper documentation that can consist on one or more than one components (hardcopies).
. physical_model	Representation as physical model (mock-up)
document_file	One file representing a document, could be a paper document, a microfilm or a CAD model.
.digital_file	Identification of a file with path and name (e.g. CATIA model name)
.hardcopy	Identification of a paper or microfilm document

Table 3: Document management objects in AP214.

A digital document represents an electronic document, while a physical document stands for a 'hardcopy', typically paper, document.

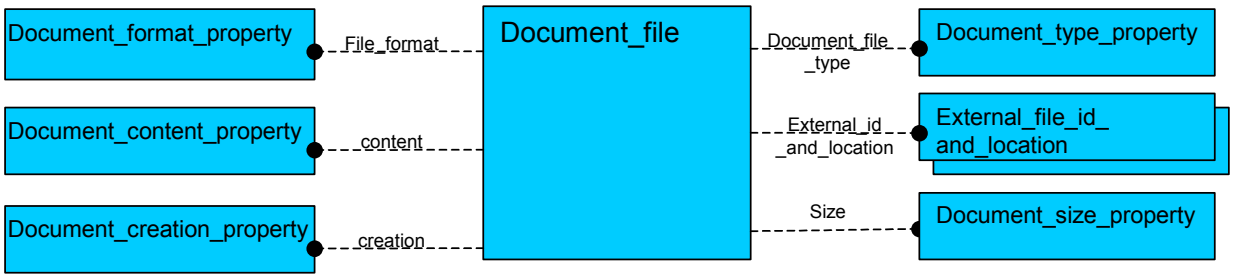


Figure 16: PDTnet Document Management.

In the definition shown in Figure 16 is no change compared with AP214.

3.10 Document Version Relationships

Using ***document_version_relationship*** AP214 allows the definition of associations among version information. The *document_version_relationship* concept is comparable with the concept for item_version_relationship described in chapter 3.2.

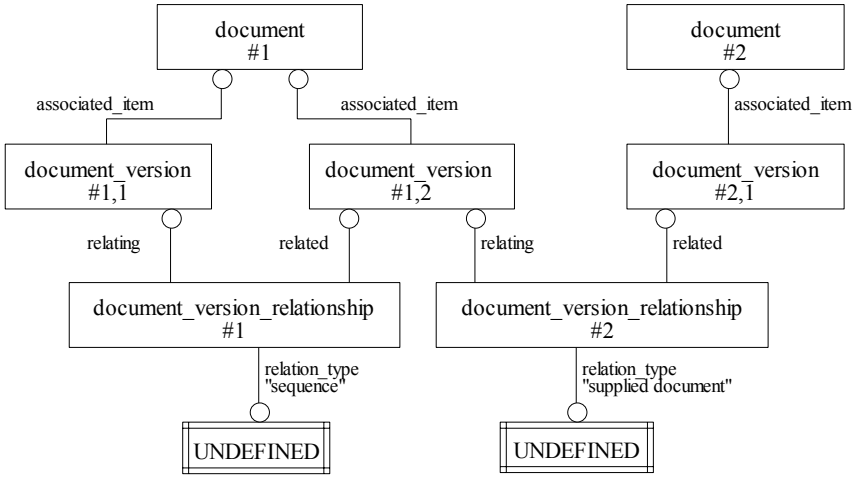


Figure 17: Document version relationships in AP214.

Regarding the definitions made in chapter 2.4 the element *document_version_relationship* was changed by deleting the attribute *relating*. In PDTnet *document_version_relationship* is defined as an element of *document_version*. The *related* attribute is used as defined in AP214.

3.11 External File

External files in AP214 represent a simple external reference to a named file using the ***external_file_id_and_location*** entity. The external file may identify a digital file or a physical, 'hardcopy' file. An external file is not managed by the system - there is no capability for managed revision control or any document representation definitions for an external file.

An external file is simply an external reference that may be associated with other product data. Document/file properties may be associated with an external file as with an identified managed document.

If a file is under configuration control, it should be represented as a constituent of a document definition view/representation according to 'Document as Product'. In this case, it is actually the managed document that is under direct configuration control; the file is, in this way, indirectly under configuration control. A change to the file results in a change to the managed document (i.e., a new version) - the changed file would be related to a new document version.

3.12 Document Properties

AP214 defines a flexible concept for defining document properties. Document_file as well as document_representation are refering with their attributes

- Document_file_type
- Content
- Creation
- File_format

- Size
- External_id_and_location

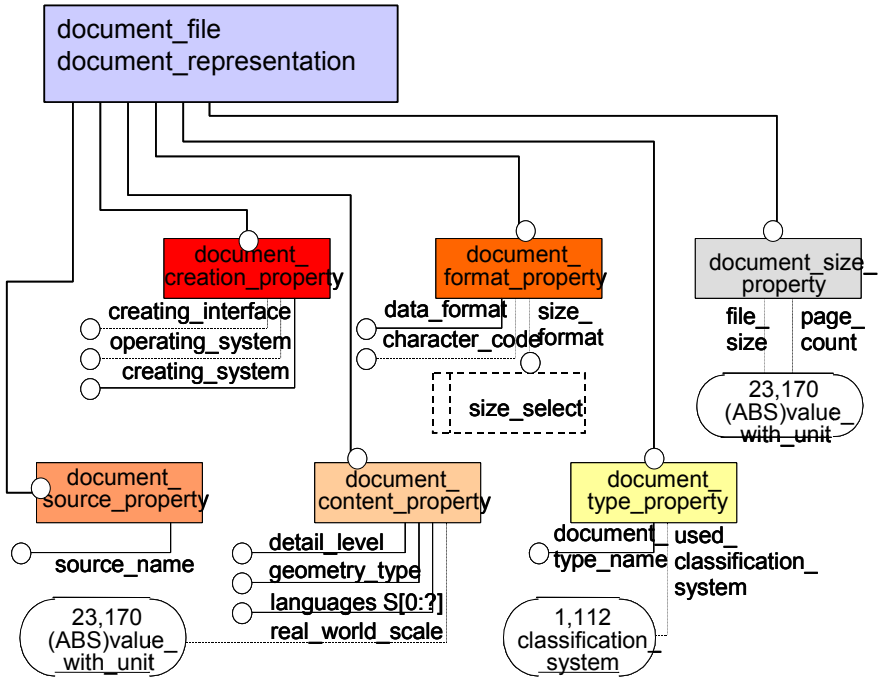


Figure 18: The AP214 document property concept

Examples for the various document properties and their values are described in below:

examples:		
document_format_property character_code	ISO LATIN-1	US ASCII 7bit
document_format_property document_format	CATIA export	PRO/ENGINEER assembly
document_content_property detail_level	rough 3D shape	other
document_content_property geometry_type	drawing derived from 3D data	assembly
document_creation_property creating_system	CATIA 4.1.8	PRO/ENGINEER V18
document_type_property. document_type_name	drawing	geometry

Table 4: Example values for document properties.

3.12.1 Document_content_property:

The document content capability specifies characteristics detailing the content of a given document object. Aspects of the content property can be modeled via the following attributes of document content:

Attribute	Definition	Value: examples and restrictions
Detail_ level	The level of detail that the document file or the document representation provides.	Where applicable the following values shall be used: • 'rough 3d shape': 3D shape model without edge rounds and fillets; • 'rounded edges': 3D shape model with edge rounds and fillets.
Geometry_type	The kind or kinds of geometry that an object contains	Where applicable the following values shall be used: • '3D wireframe model': The document contains a 3D shape model in wireframe representation; • '2D shape': The document contains a 2D shape model or contours only; • 'surface model': The document contains a 3D shape model in surface representation; • 'closed volume': The document contains a 3D shape model in closed body topological surface representation; • 'solid model': The document contains a 3D shape model in advanced boundary representation; • 'solid and surface model': The document contains a 3D shape model in surface and advanced boundary representation; • 'assembly': The document contains an assembly structure with reference to the assembled components and their transformation matrices; • 'assembly with mating elements': The document contains an assembly structure including the mating components only, such as screws or rivets, with exact positioning information. This assembly representation is intended to be overlaid with the assembly structure for the main components; • '2D drawing': The document contains a technical drawing without 3D shape representation; • 'drawing derived from 3D data': The document contains a technical drawing that has been derived from a 3D shape model; • 'drawing related to 3D data': The document contains a technical drawing that visualizes a 3D shape model and possibly establishes associative links to the 3D shape model.
languages	Language or languages are used in the characterized objects.	e.g., 'English''
Real_ world_ scale	the scale that is used	e.g., '1:50'

Table 5: Document content property.

3.12.2 Document_creation_property

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

In the case where a Document_creation_property is referred by a Document_representation the characteristics apply to all individual Document_file objects, whereas in the case it is referred by a Document_file, the characteristics apply on an individual basis.

The data associated with a Document_creation_property are the following:

Attribute	Definition	Value: examples and restrictions
creating interface	the computer application used to create the document object.	e.g., "Postscript driver"
creating system	the computer application or the machine which is used to create the object that is characterized.	e.g., 'Microsoft Word V6'
operating system	the operating system that is used to execute the computer application that created the characterized object.	e.g., 'HP-UX 11'

Table 6: Document creation property.

3.12.3 Document_format_property

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

In the case where a Document_format_property is referred by a Document_representation, the characteristics apply to all individual Document_file objects, whereas in the case it is referred by a Document_file the characteristics apply on an individual basis.

The data associated with a Document_format_property are the following:

Attribute	Definition	Value: examples and restrictions
data format	the convention that was used to structure the information in the characterized object	Where applicable the following values shall be used: 'DXF', 'IGES', 'STEP AP203', 'STEP AP214', 'TIFF CCITT GR4', 'VDAFS', 'VOXEL'
character	the character code that	Where applicable the following values shall be used:

Attribute	Definition	Value: examples and restrictions
code	is used for the stored data	'US ASCII 7bit', 'ISO LATIN-1', 'EBCDIC', 'binary'
Size_ format	the dimensions of a physical presentation	e.g., 'ISO A0', '0.2 x 0.4 x 0.4 meters'

Table 7: Document format property.

3.12.4 Document_size_property

A Document_size_property specifies the size of a Document_file or of a Document_ representation object. At least one of the optional attributes shall be specified for each instance of this object.
In the case where a Document_size_property is referred by a Document_representation, the size information is the sum of the sizes of all individual objects that are collected by this object, whereas in the case it is referred by a Document_file, the size information is the one of the individual objects that is referenced.
The data associated with a Document_size_property are the following:

Attribute	Definition	Value: examples and restrictions
file size	the value that represents the size of a digitally stored document.	2000
page count	the number of pages (of a physical document)	30

Table 8: Document size property.

3.12.5 Document_type_property

A Document_type_property specifies the kind of a Document_file.

The data associated with a Document_type_property are the following:

Attribute	Definition	Value: examples and restrictions
Document_type_ name	specifies the word or the group of words that describe the kind of object the characteristics are provided for	• 'geometry': The document represents a shape model; • 'NC data': The document represents numerical control data; • 'FE data': The document represents finite element data; • 'sample data': The document represents measured data; • 'process plan': The document represents process planning data; • 'check plan': The document represents quality

Attribute	Definition	Value: examples and restrictions
		control planning data; • 'drawing': The document represents a technical drawing.
Used_ classification_ system	specifies the Classification_system the document_type_ name is defined in.	

Table 9: Document type property.

Regarding the definitions made in chapter 2.4 the elements *document_version_relationship and document_structure were* changed by deleting their attribute *relating*. In PDTnet *document_version_relationship* is defined as an element of *document_version and* (see Figure 19). *Document_structure* is defined as an element of *document_representation*. The *related* attribute is used as defined in AP214.
In the case of document_property there is no difference to AP214.

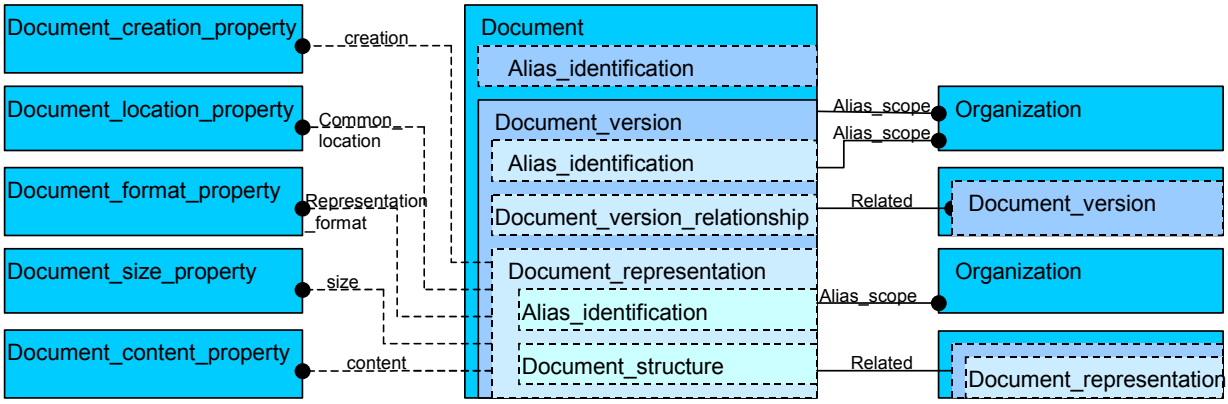


Figure 19: PDTnet instantiation of document properties.

3.13 Context Information

Context information (***application_context***) provides a scope and necessary circumstance for part identification information. The context information identifies the usage of the information within the scope of AP214.

The application domain is identified by the application_context.application_domain attribute while the lifecycle stage is represented with the attribute application_context.life_cycle_stage.

4 Conclusion

The chapters above explained the usage of PDTnet constructs based on AP214 definitions. All examples showed that differences between AP214 and PDTnet can be managed by a view simple rules.
The PDTnet Usage Guide will be extended by request of application projects.