PDM
Implementor Forum

Recommended Practices for
STEP PDM Schema
February 2000

Contacts:

Jochen Boy
PROSTEP AG
Taunusstraße 11, D-80807
München
+49 89 35020 210
jochen.boy@prostep.com

Phil Rosche
PDES, Inc.
5300 International Blvd.
North Charleston, SC 29418
843-760-3593
rosche@aticorp.org

Usage Guide for the STEP PDM Schema

Figures

Instance Diagrams

Exchange File Examples

Abstract

The STEP PDM (Product Data Management) Schema is a reference information model for the exchange of a central, common subset of the data being managed within a PDM system. It represents the intersection of requirements and data structures from a range of STEP Application Protocols, all generally within the domains of design and development of discrete electro/mechanical parts and assemblies.

The STEP PDM Schema is *not* a specification for the functionality required for the complete scope of all PDM system functionality – i.e., it is *not* the union, but the intersection, of functionality present in the set of STEP Application Protocols. There exists functionality important for complete PDM functionality that is not represented in the PDM Schema, but in other units of functionality present in STEP APs.

By definition, a PDM system is something that manages data about products. At the central core of PDM information is product identification. A product in STEP represents the concept of a general managed item within a PDM system. In the STEP PDM Schema, the general product concept may be interpreted as either a Part (see section 1) or a Document (see section 5). In this way, parts and documents are managed in a consistent and parallel fashion. Section 12 describes a mechanism to associate product data with an additional identifier (alias).

Also central to the functionality of many PDM systems is identification of external files (both digital and physical), their relationship to managed documents (see section 8), and how they can be associated with core product identification (see section 10). The external file reference mechanism in the STEP PDM Schema is described in section 7 of this document.

Classification of products is important in a PDM system for information classification and retrieval. It also supports basic type distinction between products that are parts and those that are documents. In the PDM Schema, product classification is used consistently for parts (see section 2) and documents (see section 6).

Product properties are integrally related to the definition of an identified product, and so are naturally also included in the central core PDM information. Sections 3 and 9 discuss properties associated with an identified product, interpreted, respectively, as either a part or a document.

Various general authorization and organizational data that are related to core product identification play an important role in PDM systems. Section 13 of this document describes the various organizational and management constructs that support product authorization in the STEP PDM Schema.

Product structures are the principle relationships that define assemblies and product configurations. Section 4 details part structures in the STEP PDM Schema; section 11 describes document structures. Configuration identification and effectivity information related to these structures is detailed in section 14.

Section 15 describes structures to manage the documentation of requests and corresponding orders for engineering action, in support of the change management process. Also included are representations for contract and project identification.

Finally, Section 16 summarizes recommendations related to measures and units.

Acknowledgements

Anna Wasmer
Glen Ziolko

Overview

Usage guide goal: to describe the recommended structure and attribute population for particular instance models created from the EXPRESS entities and types defined by the STEP PDM Schema. The selected instance models illustrate how to encode data values that need to be exchanged in support of key industry requirements common across the product manufacturing domain.

Usage guide status: Release 4.1 covers the entire scope of the PDM Schema . The maintenance of this document is guided by the PDM Implementor Forum where issues and clarifications are raised and resolved. Issues may be put forward to the points of contact identified on the cover page of this document.

Intended audience: developers of applications and information management systems that must use product data and exchange it with other systems and applications in support of the business processes related to product design and development. Anyone interested in the scope of the requirements supported by the STEP PDM Schema.

Intended use: manual and companion to the developer of STEP data exchange and translator software used by applications and information management systems that rely on product data. Guideline for consistent preprocessor instance model creation and requirement value encoding to enable meaningful information exchange between different systems and applications using STEP. Guideline for consistent interpretation by a postprocessor of a STEP Part 21 exchange file according to the unified STEP PDM Schema.

Usage guide style: overall document proceedes in an incremental, step-by-step fashion to describe, and in parallel to illustrate, the recommended instantiation of the EXPRESS entities and types in the PDM Schema.

The diagram figures are presented using a graphical notation intended to illustrate the instance model. This notation *is not* EXPRESS-G and *does not* illustrate the EXPRESS schema; rather it is a graphical illustration of a specific population of a particular instance model of the schema. This notation supports:

- illustration of entity instances,
- illustration and identification of referenced instances that are not fully illustrated in the current figure,
- illustration and identification of the multiple possible referenced instances corresponding to an attribute that has a select type as the value,
- indication of optional attributes and optional structure (dashed lines),
- illustration and identification of groups of functionally related instances (shaded bounding box),
- identification of specific attribute values (typically string values, maybe enumerated type values).

Each instance diagram figure is accompanied by a related STEP Part 21 exchange structure example. The example exchange file corresponds directly to the related instance model diagram and illustrates the very same thing using a different notation, i.e., STEP Part 21 syntax versus the graphical instance model notation.
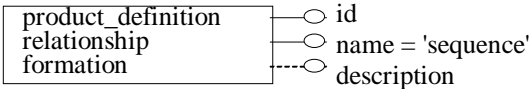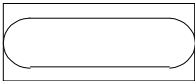
Usage guide structure: the overall scope of requirements is partitioned into a set of major sections corresponding to identified units of functionality. Within a major section, there may be sub-sections. These sub-sections further divide the scope into smaller components of coherent functionality that interact with each other to realize the functionality of the entire unit.

There is generally a description of requirements and a corresponding instance diagram associated with each section and sub-section of this document. Each instance diagram is followed by a detailed explanation and specific recommendations for the EXPRESS entities used in the instantiation diagram example. The entity listing and explanation is in turn followed by the corresponding STEP Part 21 exchange structure example.

Within a section, diagrams corresponding to sub-sections incrementally build upon one another to finally achieve a complete instance model example that illustrates the entire scope of the unit of functionality.

General Information

Instantiation diagrams: the diagrams are presented using a graphical notation intended to illustrate the instance model. This notation is not EXPRESS-G, however it does intentionally resemble it. The diagrams do not illustrate the EXPRESS schema, but illustrate a specific population of a particular instance model (or portion thereof) of the schema. A legend for the diagram notation is shown below :

- illustration of entity instances
  and their attributes;

- identification of optional attributes;

- indication of specific attribute values,
  typically strings, may be enumerated values;

  ```
  product_definition        ──○ id
  relationship              ──○ name = 'sequence'
  formation                 ----○ description
  ```

- identification of other document sections
  and referenced entity instances
  not fully illustrated in the current figure;

Entities and attributes not supported by the preprocessor: for various reasons, there may be some entities that cannot be completely exported by the preprocessor. Sometimes an application may not maintain all the information that is anticipated for the data exchange. Other times, the information may be maintained by a sending system but not included in the data exchange. Never the less, the preprocessor must provide values for all mandatory attributes in an exchange file. For mandatory string attribute values, the null (empty) string '' has often been used when a preprocessor can provide no real user data. The default string value '/NULL' may be used for this purpose, as recommended by the European automotive industry. When no data is provided by a sending system for a string value, the preprocessor should use '/NULL' or the empty string ''. To further indicate the reason why no data is provided, the following convention may be used:

- Empty string '' indicates user data managed by the sending system but not provided for data exchange.
- String '/NULL' indicates user data in a mandatory attribute that is not managed by the sending system or currently not known.
- $ is used in the physical file, if an optional attribute is not instantiated.

It is generally not recommended to use the empty null string '' or the default string '/NULL' as valid user data.

Entities and attributes not supported by the postprocessor: for various reasons, there may be some entities that cannot be completely imported by the postprocessor.   The postprocessor translator implementation simply may not support the import of the entity. The receiving system may not maintain the information that is carried by an entity or attribute, or it may require specific attribute values that are not present in the input data. Entities and attributes not imported should list a reason in a history log file. Entities and attributes not supported by the receiving system should not cause a system failure. The minimum acceptable behavior should be to ignore the unsupported constructs.

Unspecified and optional attribute values: optional attributes without specific recommended values, such as the description attribute, are available on many entities in the PDM Schema. In general, use of this type of attribute is given the following recommendation:

Preprocessor - first, follow the usage guide as much as is possible - if some specific common harmonized user requirement has been documented in the usage guide for the attribute, try to adapt this requirement to

those you have identified (i.e., map the standard into your user domain). If no specific common harmonized user requirement has been documented in the usage guide, in general such an optional attribute should not be instantiated. However, these attributes may be used in some bilateral agreements between exchange partners.

Postprocessor - any optional attribute with no specific mapping specified in general cannot be specifically interpreted in an interoperable way. While these types of attributes are in general not recommended to be instantiated, the postprocessor should gracefully handle any data that is exchanged using these attributes. A robust, interoperable PDM Schema processor will generally provide user access to the values exchanged.

Derived attributes: in general, derived attributes are not given with the description and recommendations for entities in the PDM schema. This is consistent with the STEP part 21 specification where derived attributes are not represented in an exchange file. Only in certain cases where special attention is required will such an attribute be presented and explained in this usage guide.

Implementation project specific values: attribute values recommended in this usage guide should be supported by systems conforming to the PDM Schema. Other values negotiated between exchange partners in specific projects may be used where the interpretation of their meaning does not contradict definitions provided in this usage guide. However, these agreements will not generally be interoperable solutions.

Leading and trailing blanks in STRING values: all white space within the single quote delimiters of a STRING value should be considered valid user data.

Uniqueness of identifiers: Identifiers cannot be unique in general. In the parallel management of internal and external identifiers in a database, duplicate identifiers may occur that can cause problems with the uniqueness rules defined in the EXPRESS schema. In those cases, the interpretation of the identifier can be logically extended by a prefix that identifies the organizational scope (id value of the organization related in role 'id owner') to help ensure uniqueness (see1.1.1.1). The unique rule shall be evaluated only in scope of the organization defined by the id_owner. Thus, a processor may utilize the organization identifier as a key to identifying a product.

Schema version identification: version identification for the PDM Schema shall be encoded in the header section of the STEP Part 21 exchange file to identify the version of the schema to which the file conforms. This is done with the header entity *file_schema*, which identifies the EXPRESS schemas that specify the entity instances in the data section. The attribute *schema_identifiers* contains a list of strings that name the schema, optionally followed by the object identifier assigned to that schema. In place of the object identifier, the PDM Schema version identification number shall be enclosed within curly braces.
Only capital letters shall be used in schema name strings.

EXAMPLE: To indicate PDM Schema version 1.1, the following instance of the header entity file_schema should be used.

```
ISO-10303-21;
HEADER;
...
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
```

Scope of the STEP PDM Schema

The STEP PDM Schema is a core set of entities in STEP that support the mapping of concepts for Product Data Management (PDM). This document and the PDM Schema are the result of a cooperative development process between ProSTEP and PDES, Inc. The PDM Schema has been established to promote interoperability between STEP APs in the area of product data management.

☞ **Common PDM data Schema generated and maintained by PDES, Inc. and ProSTEP**

☞ **Real Subset of PDM relevant STEP APs (AP203, 212, 214, 232)**

☞ **Fulfills nearly all requirements for PDM data exchange**
Main functionality for parts and documents:
- identification
- versioning
- structures incl. transformations
- approvals and authorization
- project, work order, work request
- effectivities
- classification and properties

Figure 1: Positioning and Contents of the PDM Schema

This document summarizes and connects with implementation experiences the PDM Schema, version 1.1. It introduces a set of functional areas that together describe the scope of functionality of this release of the PDM Schema. These units of functionality suggest a modular structure for the PDM Schema.

Units of Functionality

The product data management requirements addressed by the PDM Schema are organized into groupings of related concepts. These groups provide a logical grouping of AIM entities for the purpose of a clear structure within the PDM Schema. These groups suggest a modular structure for the PDM Schema in line with the current direction towards modularization of the technical contents of STEP integrated resources and Application Protocols.

The modular semantic units of functionality are listed below:

- Part Identification,
- Part Classification,
- Part Properties,
- Part Structure and Relationships,

- Document Identification,
- Document Classification,
- External Files,
- Relationships Between Documents and Constituent Files
- Document and File Properties,
- Document and File Association to Product Data,
- Document and File Relationships,

- Alias Identification,

- Authorization,

- Configuration and Effectivity Information,

- Work Management Data.

1   Part Identification

The PDM Schema manages all parts as products, according to a fundamental STEP interpretation of 'Part as Product'.   Part identification is achieved using basic product identification.   The identified product may represent a part, a tool, or raw material.   In addition, a product may represent a managed document and be identified according to the 'Document as Product' interpretation (see Section 5.1).

Part identification is the center for assignment of further product management data. As a consequence, at least one product identification (part or document) must be instantiated for the exchange of part/document data in the STEP PDM Schema.

The simple alias concept supports assignment of an alternate identifier to a product. This mechanism may also be used to alias other elements of product data.

Do not use the alias identification for the requirement of supplied part or document identification. This more complex alias relationship concept is specific for identification and renumbering of supplier's parts and documents.

1.1   Part as Product

The PDM Schema manages all parts as products. Identification of products in the STEP PDM Schema consists of three distinct concepts:

- Product Master Identification,
- Context Information,
- Type Classification.

Product master identification maintains the base part number, distinct part version identification, and information identifying a view definition. An identified part may be a single piece part or an assembly of arbitrary complexity.

Context information provides a scope and necessary circumstance for interpretation of product identification information.

Type classification distinguishes products representing parts from those that represent documents. Further possible type classifications like 'tool' or 'raw material' will be described in a future release in separate sections, e.g., to describe the relationship between a part and its raw material.

1.1.1   Product Master Identification

Product master identification supports the ability to uniquely identify a part. The identification of products in the STEP PDM Schema consists of three important and structurally distinct concepts:

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

The master base identification maintains information common to all product versions and disciplines and/or life cycle views. It contains the base product number and name. The base number should not be subject to any encoding of information into a single complex parseable string.

There must be at least one version assigned to a product 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 product master. The set of versions associated with a base product 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.

It is recommended that at least one view definition be assigned to each product version. There may be two exceptions to this general rule:

- Supplied product identification, in which case a supplied product may be represented by only product base;
- Version identification and when version history is represented, where only the most recent current version is required to have an associated view definition.

The product definition view collects information relevant from the perspective of a particular application domain or life cycle stage. There may be more than one life-cycle view definition associated with a particular version of a product. This is especially important to enable different views on product assembly structures.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of part master identification are illustrated in Diagram 1 below. The corresponding STEP exchange file encoding is given in Example 1.
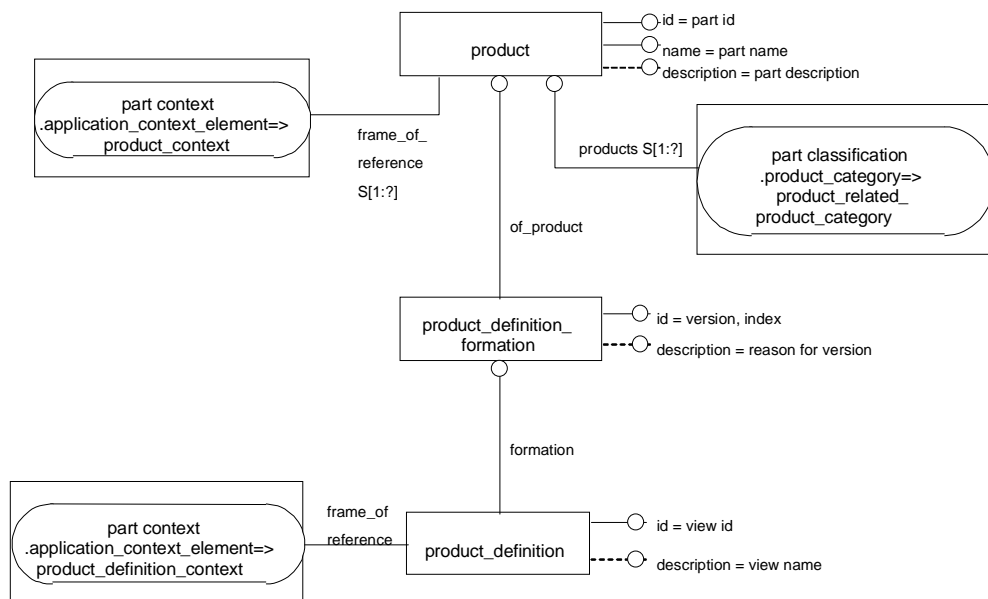
Diagram 1: Part Master Identification Instance Diagram

1.1.1.1 product

The product entity represents the product master base information. This entity collects all information that is common among the different versions and views of the product. The product number is strictly an identifier. It should not be used as a 'smart string' with some parseable internal coding scheme, e.g., to identify version or classification information.

The product number identifier must be unique within the scope of the business process of the information exchange. This is typically not a problem when the product 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., product.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 product.id attribute is not unique within the domain of the business arrangement of the exchange.

Attributes
- The *id* attribute stores the unique base identifier for a product, the product number.
- The *name* attribute stores the nomenclature, or common name, of the product.
- The *description* attribute should contain an expanded name or description of the product.
- Context information is stored in the *frame_of_reference* attribute. All products in STEP must be founded in some product_context. This requirement is discussed in more detail in 1.1.2.

ENTITY product	Attribute Population	Remarks
id	type: identifier = string product number identification	Unique within the scope provided by organization in role 'id owner'
name	type: label = string	
description	type: text = string	
frame_of_reference	type: entity = product_context	SET [1:?]

Preprocessor Recommendations: All preprocessors should use valid user (non-defaulted) data for the values assigned to the attributes id and name, as well as to the id of the organization related in the role 'id owner'. In populating the id attribute, the identifier must be unique within the scope of the related organization or person_and_organization in the role 'id owner'.

Postprocessor Recommendations: Postprocessors should provide user access to the values of the attributes id and name, as well as the id of the organization related in the role 'id owner'.

Related Entities: The PDM Schema requires that all products exist in at least one product_category, to provide type classification information. This type classification requirement is discussed in 1.1.3.

It is strongly recommended that products have a related organization or person_and_organization that identifies responsibility for the original design of the part. The role of this assignment is identified as the 'id owner'. This is the organization or person_and_organization that defines the part and assigns the part number. See 13.1 for guidance on how to create the organization or person_and_organization entities.

In addition, the product may optionally be referenced by the following entities:

- applied_organization_assignment (or applied_person_and_organization_assignment) to represent the customer, or other associations of an organization to a product (see 13.1);
- product_related_product_category to classify a product with respect to specific criteria. One product_related_product_category is required to represent product type (see 1.1.3).

1.1.1.2 product_definition_formation

The product_definition_formation entity represents the identification of a specific version of the base product identification. A particular product_definition_formation is always related to exactly one product.

Each instance of product is required to have an associated instance of product_definition_formation. A single product entity may have more than one associated product_definition_formation. The set of these product_definition_formation entities represents the revision history of the product.

Attributes
- The *id* attribute uniquely identifies this version of the related product.
- The *description* attribute contains the reason for the creation of the version.
- The attribute *of_product* provides a link to the base product of which this entity represents a version.

ENTITY product_definition_formation	Attribute Population	Remarks
id	type: identifier = string part version identification	Unique in relation to the specified of_product
description	type: text = string	OPTIONAL
of_product	type: entity = product	

Preprocessor Recommendations: If an organization does not version parts, it is recommended that the id attribute contain the string '/NULL' to indicate that no version information is relevant or intended. In this case only a single product_definition_formation for the part is possible. Some preprocessors have used an id value of '/ANY' to indicate that *any* existing revision of a component is valid for the parent assembly. This technique may reduce the amount of data sent in change packages, but it also reduces the ability to track the actual contents of parts lists at a particular change level.

Postprocessor Recommendations: If the value of the id attribute for a product_definition_formation is the string '/NULL', postprocessors should use this as an indication that the sending system or business process does not support versioning of parts. Postprocessors may also recognize an id value of '/ANY' as a generic revision of a part when it is involved as a component in an assembly. This has been used to indicate that *any* existing revision of the component is valid for use in the parent assembly.

Related Entities: In general, each product_definition_formation is recommended to have an associated product_definition representing the view definition for the part version. In restricted cases, a part version without a definition may be used to enhance information about another related, fully defined version. In two specific cases a part version may be exchanged without an associated view definition:

1. When version history (sequence relationship) is represented - only the most recent version is required to have an assigned product_definition. If there is no product_definition associated with the previous versions, only basic information about the sequence of previous versions is exchanged as additional information about the current part version that is the focus of the data exchange;
2. Where a supplied item is identified using the alias relationship - the supplier part version may not have an associated product_definition.

In addition, a product_definition_formation may be optionally referenced by the following entities:

*   product_definition_formation_relationship to associate two product versions with each other to characterize the version history;
*   applied_organization_assignment (or applied_person_and_organization_assignment) to represent the update/modifier, customer, or other associations of an organization to a product version;
*   applied_date_assignment (or applied_date_and_time_assignment) to represent various dates related to a product version;
*   applied_approval_assignment to record the authorization status of a product version (see 13.2);
*   applied_security_classification_assignment to record a level of security clearance for a product version.

1.1.1.3 product_definition_formation_with_specified_source

The product_definition_formation_with_specified_source entity is a subtype of the entity product_definition_formation. This entity adds the attribute make_or_buy to indicate the source of the version - either 'made' or 'bought'. This entity is never used in the context of 'Document as Product', but it may be used in 'Part as Product' for compatibility with AP203. However, this make-versus-buy distinction can be ambiguous in the context of exchange - going down a supply chain, the sender is often 'bought' while the receiver is 'made', while in the other direction, the sender is 'made' and the receiver 'bought'. Because of this ambiguity the use of this entity is not generally recommended.

Attributes
*   The *make_or_buy* attribute contains the source information.

ENTITY product_definition_formation_ with_specified_source	Attribute Population	Remarks
id	Same as supertype.	Same as supertype.
description	Same as supertype.	Same as supertype.
of_product	Same as supertype.	Same as supertype.
make_or_buy	type : source (enumeration)	not recommended

Preprocessor Recommendations: The use of this subtype is optional. It is not recommended in the general case. The source value is an enumerated type having possible values '.MADE.', '.BOUGHT.', or '.NOT KNOWN.'. '.MADE.' indicates the part is built within the company. '.BOUGHT.' indicates a vendor part, but this distinction can be unclear when the data is exchanged.

Postprocessor Recommendations: If this entity is encountered, the postprocessor should at a minimum process the information from the supertype entity product_definition_formation.

Related Entities: There are no specific related entities.

1.1.1.4 product_definition

The product_definition entity represents the identification of a particular view on a version of the product base identification relevant for the requirements of particular life-cycle stages and application domains. View may be based on application domain and/or life cycle stage (e.g., design, manufacturing). A view collects product data for a specific task. It is possible to have many product_definition views for a part/version combination.

The product_definition entity enables the establishment of many important relationships. Product_definition is the central element to link product information to parts, e.g., assembly structure, properties (including shape), and external descriptions of the product via documents.

The use of product_definition entities is not strictly required by rules in the PDM Schema, but it is strongly recommended. All product_definition_formation entities should always have at least one associated product_definition, except in the case of supplied product identification and version history information.

Attributes
- The *id* attribute identifies which view of the product the particular instance represents.
- The *description* attribute specifies the word or group of words used to refer to the product_definition.
- The *formation* attribute links to the product_definition_formation of which this represents a view.
- Context information is stored in the *frame_of_reference* attribute. All STEP product_definitions must be founded in some product_definition_context that identifies the application domain and life cycle stage from which the data is viewed (see 1.1.2.).

ENTITY product_definition	Attribute Population	Remarks
id	type: identifier = string	must be unique in relation with a specific product_version
description	type: text = string	OPTIONAL
formation	type: entity = product_definition_formation	reference to the associated product_definition_formation
frame_of_reference	type: entity = product_definition_context	reference to the associated product_definition_context (see 1.1.2.4)

Preprocessor Recommendations: There is no standard mapping for the id attribute of product_definition, however the value should be unique relative to others related to the same product_definition_formation. Previous use of the id attribute on product_definition had sometimes 'overloaded' the unique identifier with

informal life cycle or organization information. This is not generally recommended for the PDM Schema - this attribute should contain a unique identifier for the part view definition - no additional semantics are associated with this attribute in the PDM Schema.

Postprocessor Recommendations: Previous use of the id attribute on product_definition had sometimes 'overloaded' the unique identifier with informal life cycle or organization information, this should not be expected in general or required for postprocessing.

Related Entities: A product_definition may be referenced by the following entities:

- applied_identification_assignment to assign alias identifiers,
- product_definition_relationship to associate two product_definitions with each other to characterize various product structures,
- property_definition to associate general properties,
- configuration_design to associate configuration identification information,
- applied_organization_assignment (or applied_person_and_organization_assignment) (see 13.1),
- applied_date_assignment (or applied_date_and_time_assignment) to represent various dates related to a product version,
- applied_approval_assignment to record the authorization status of a product version (see 13.2),
- applied_security_classification_assignment to record a level of security clearance for a product version,
- applied_document_reference to assign documents (external descriptions) to the product_definition.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#30 = PRODUCT_CONTEXT('', #20, '');
#40 = PRODUCT('part_id', 'part_name', 'part_description', (#30));
#60 =
PRODUCT_DEFINITION_FORMATION('pversion_id','pversion_description',
#40);
#80 = PRODUCT_DEFINITION('view_id', 'view_name', #60, #90);
#90 = PRODUCT_DEFINITION_CONTEXT('part definition', #20, '');
```

Example 1: exchange file segment for part master identification

1.1.2 Context Information

Context information provides a scope and necessary circumstance for product identification information. It consists of two separate and related areas:

- Application Protocol Identification,
- Application Context Information.

Application Protocol identification is provided by the entity application_protocol_definition. It characterizes the STEP Application Protocol (AP) or similar working draft STEP specification that includes or uses the PDM Schema and provides the scope or general context for the exchange data set. The general context identifies the usage of the information within the scope of the PDM Schema, and may define the application domain which provides a basis for the interpretation of all information represented in the product data exchange. It is generally recommended that application protocol identification and general context information be handled structurally the same for 'Part as Product', 'Document as Product' as for 'Product Concept'. A single instance of the entity application_context is typically referenced by all product_context and product_concept_context entities. This application_context instance is referenced by a single instance of the entity application_protocol_definition.

Application context information identifies the usage of the information within the scope of the PDM Schema. Application context information is divided into application domain information and product definition context information:

- The application domain is identified by the application_context entity. Each application domain is represented by an instance of application_context that is referenced by product view definitions (product view and type definition) which belong to the application domain;
- The product_definition_context carries information distinguishing the life-cycle stage (e.g., design, manufacturing) relevant to a particular product view definition as well as indication of the type of the definition - e.g., part definition, functional definition.

Identification of a valid scope and context of interpretation for the identifier of a product is also required in the PDM Schema. This involves the assignment of an organization to the product, in the role 'id owner' to identify the scope of uniqueness and validity of the product id.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of context information related to basic product identification are illustrated in Diagram 2 below. The corresponding STEP exchange file encoding is illustrated in Example 2.
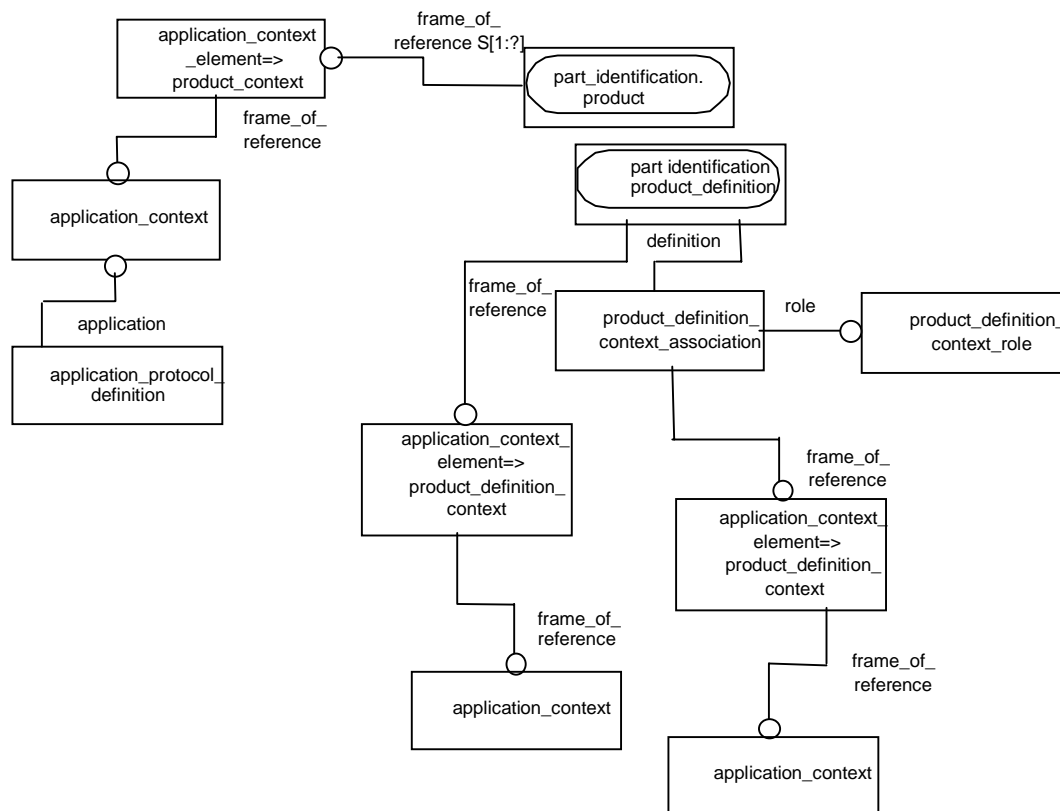
Diagram 2: Part Master with Context Information Instance Diagram

1.1.2.1 application_protocol_definition

The application_protocol_definition entity represents the identification of the STEP Application Protocol (AP) that specifies the scope and extent of the application domain appropriate for this information model. An AP that uses the PDM Schema should provide values for the attributes of this entity.

Attributes
- The *application_interpreted_model_schema_name* attribute identifies the name of the EXPRESS schema that specifies the information model used for the data representation and exchange.
- The *application_protocol_year* attribute identifies the year of publication of the specification.
- The *status* attribute provides information about the degree of maturity of the defining specification.
- The *applications* attribute provides a reference to the associated application_context entity.

ENTITY application_protocol_definition	Attribute Population	Remarks
application_interpreted_model_schema_name	type: label = text 'pdm_schema'	If the PDM Schema is contained within an Application Protocol, the AP name should be used.
application_protocol_year	type: year_number = integer 1998	If the PDM Schema is contained within an Application Protocol, the AP year should be used.
status	type: label = text 'version 1.1'	If the PDM Schema is contained within an Application Protocol, the AP status should be used.
applications	type: entity = application_context	

Preprocessor Recommendations: This entity should be instantiated exactly once in the data set. In some cases, an application protocol may support multiple application_contexts. The entity application_-protocol_definition should reference the 'primary' defining application context for the product data that is being exchanged.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

1.1.2.2 application_context

The application_context entity identifies the application domain that defined the data. The application_context entity may have an identifier associated with it through the entity id_attribute and its attribute_value attribute. The application_context entity may have a description associated with it through the entity description_attribute via the attribute attribute_value. It is not recommended to instantiate these optional values.

In the case of application protocol identification the application domain is optional, which provides a basis for the interpretation of all information represented in the product data exchange.

In the case of application context information there exists a 'primary' application context for each product_definition, this is the value of the attribute product_definition.frame_of_reference; it is the frame_of_reference for the 'primary' product_definintion_context (see 1.1.2.4). This 'primary' application context represents the defining application domain for each product_definition. Additional application domains may be associated with a product_definition through additional product_definition_contexts via the entity product_definition_context_association.

Attributes
- The *application* attribute identifies the name of the general application domain that defined the data.

ENTITY application_context	Attribute Population	Remarks
application	type: label = text	

Preprocessor Recommendations: This entity should be instantiated once in the data set for each relevant application domain.   There is no standard mapping for the application attribute of application_context in the case of application protocol identification. Therefore application domain is optional in this case.

Where appropriate, the value 'assembly study', 'digital mock up', 'process planning', 'electrical design', or 'mechanical design' should be used. Note that 'electrical design' and 'mechanical design' are not understood to mean the design life cycle - e.g., within the application domain 'mechanical design', you may have concurrent 'manufacturing' and 'design' life-cycle view definitions.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

1.1.2.3 product_context

The product_context entity is a subtype of application_context_element. All STEP products must be founded in a product_context to specify the point of view on the application. The product_context entity identifies the engineering discipline's point of view from which the data is being presented. This entity will establish the context perspective and source of requirements for product entities (see 1.1.1.1).

Attributes
- The *discipline_type* attribute contains a description of the discipline point of view for a product.

ENTITY product_context	Attribute Population	Remarks
name	type: label = text	
frame_of_reference	type: entity = application_context	
discipline_type	type: label = text	

Preprocessor Recommendations: It is recommended to instantiate this entity exactly once in the data set. There is no standard mapping for the name attribute of a product_context. There is no standard mapping for the discipline_type attribute.

Postprocessor Recommendations: Past use of the attribute discipline_type may result in values such as 'mechanical' or 'electrical'.

Related Entities: There are no specific related entities.

1.1.2.4 product_definition_context

The product_definition_context entity is a subtype of application_context_element.   All STEP product_definitions must be founded in a product_definition_context to identify the product definition type and life cycle stage from which the data is viewed.

There are two ways to implement the product_definition_context in STEP:

- association of a single primary context with a product_definition,
- assignment of more than one (additional) context to a product_definition.

The value of the attribute product_definition.frame_of_reference is a product_definition_context that represents the 'primary' defining context for a view.   In general this defining context is qualified both by

type (product_definition_context.name) and by life-cycle (product_definition_context.life_cycle_stage) information. The primary context also has associated application domain information (see 1.1.2.2).

The product_definition_context_association entity allows for multiple additional contexts to be associated with a single product_definition. There is always one required 'primary' context that identifies the defining application domain and life cycle information. Additional product_definition_context entities identify additional concurrent relevant views on the product_definition. These application_contexts are related to the product definition by product_definition_context_association.

Attributes
- The *life_cycle_stage* attribute identifies the life-cycle view on the associated product_definition.

ENTITY product_definition_context	Attribute Population	Remarks
name	type: label = text	Distinguishes the type of the associated product_definition
frame_of_reference	type: entity = application_context	
life_cycle_stage	type: label = text	string appropriate to describe the life-cycle point of view

Preprocessor Recommendations: The name attribute provides a distinction on the type of view on a part version ('part definition') from one of a document version ('digital document definition', 'physical document definition'). This attribute may also indicate other types of definitions: e.g., functional, or spatial and/or zonal.

The attribute life_cycle_stage contains a description of the particular viewpoint from which the data may be interpreted. Recommended values include 'design' and 'manufacturing'. Preprocessors should identify life-cycle information for the defining 'primary' product_definition_context, they may also indicate additional relevant life_cycle_stage information with a second product_definition_context, related via the product_definition_context_association

Postprocessor Recommendations: Postprocessors should interpret the value of the name attribute as a type distinction between various definitions of parts and documents. The life_cycle_stage attribute value may be interpreted as the relevant viewpoint from which the data is valid. Interoperable processors will understand life-cycle information from the defining 'primary' product_definition_context as well as any additional life-cycle information provided by other relevant product_definition_context entities related via the product_definition_context_association. The product_definition_context_association entity may or may not be present relating a product_definition with the 'primary' context information.

Related Entities: There are no specific related entities.

1.1.2.5 product_definition_context_association

The product_definition_context_association entity allows for multiple additional contexts to be associated with a single product_definition.

With 'Part as Product', there is always one required 'primary' context that identifies the defining application domain and life cycle information from which the data is viewed. This 'primary' context is the value of the attribute product_definition.frame_of_reference. Additional product_definition_context entities identify additional concurrent relevant views on the product_definition. These additional contexts are related to the product definition by the entity product_definition_context_association.

Attributes
- The *definition* attribute provides a reference to the associated product_definition entity.
- The *frame_of_reference* attribute is a pointer to the associated product_definition_context entity.
- The *role* attribute gives an optional role indication to the association.

ENTITY product_definition_context_ association	Attribute Population	Remarks
definition	type: entity = product_definition	
frame_of_reference	type: entity = product_definition_context	
role	type: entity = product_definition_context_role	

Preprocessor Recommendations: There are no specific preprocessor recommendations

Postprocessor Recommendations: Interoperable postprocessors should expect that a product_definition_context_association entity might be, or might not be, present to relate a product_definition with its 'primary' context information.

Related Entities: There are no specific related entities.

1.1.2.6 product_definition_context_role

The product_definition_context_role entity provides a role string that is related to a product_definition_context_association entity.

Attributes
- The *name* attribute provides the word or group of words by which the role is referred.
- The *description* attribute provides additional descriptive information related to the role.

ENTITY product_definition_context_role	Attribute Population	Remarks
name	type: label = string	'additional context'
Description	type: text = string	OPTIONAL

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#10 = APPLICATION_PROTOCOL_DEFINITION('version 1.1','pdm_schema',1998,
#20);
#20 = APPLICATION_CONTEXT('mechanical design');
#30 = PRODUCT_CONTEXT('', #20, '');
#40 = PRODUCT('part_id', 'part_name', 'part_description', (#30));
#60=PRODUCT_DEFINITION_FORMATION('pversion_id','pversion_desc',#40);
#80 = PRODUCT_DEFINITION('view_id', 'view_name', #60, #90);
#90 = PRODUCT_DEFINITION_CONTEXT('part definition', #20, '');
```

Example 2 : exchange file segment for part master with context information

```
#10 = APPLICATION_PROTOCOL_DEFINITION('version 1.1','pdm_schema',1998,
#20);
```

```
#20 = APPLICATION_CONTEXT('');
#30 = PRODUCT_CONTEXT('', #20, '');
#40 = PRODUCT('part_id', 'part_name', 'part_description', (#30));
#60 = PRODUCT_DEFINITION_FORMATION('pversion_id','pversion_desc',#40);
#80 = PRODUCT_DEFINITION('view_id', 'view_name', #60, #90);
#90 = PRODUCT_DEFINITION_CONTEXT('part definition', #100, 'design');
#100 = APPLICATION_CONTEXT('mechanical design');
#110 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION (#80, #130, #120);
#120 = PRODUCT_DEFINITION_CONTEXT_ROLE ('additional context', $);
#130 = PRODUCT_DEFINITION_CONTEXT('', #100, 'manufacturing');
```

Example 3: exchange file segment for part master without application domain of application protocol identification and multiple application context information

1.1.3 Type Classification

Type Classification information provides the basic capability to distinguish between products interpreted as *parts* and those interpreted as *documents*.   This capability also supports distinguishing between different types of parts, e.g., *detail*, *assembly*, or *standard* parts.

A 'detail' is a 'part' - it represents a part that is not broken down into other identified, managed component parts in a product structure. The value 'detail' is used to distinguish from 'assembly' if this is required when exchanging product structure.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of part type classification are illustrated in Diagram 3.
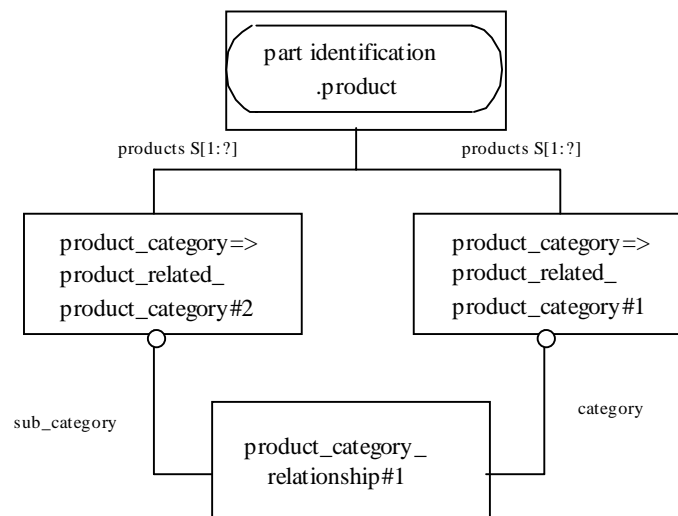
Diagram 3 : Part Master with Part Type Instance Diagram

1.1.3.1 product_related_product_category

The product_related_product_category entity is a subtype of product_category.   It represents the identification of a specific classification applied directly to a product. The name and description attributes

are inherited from the supertype product_category. This subtype adds the attribute products that allow it to be associated (related) directly to a product instance.

Note: This classification relates to the process view on the product. It is not an indication whether or not the associated information is actually given in the file. For example a product might be classified as an assembled part despite the fact that the information on the product structure related to this assembly is not given in the file.

Attributes
- The ***products*** attribute associates the category with the product entity instances to which it applies.

ENTITY product_related_ product_category	Attribute Population	Remarks
name	type: label = string 'part', 'detail', 'assembly', 'standard'	The primary value to use is 'part' - the values 'detail', 'assembly', and 'standard' indicate a further type distinction on the part.
description	type: text = string	OPTIONAL
products	type: entity = product	SET[1:?]

Preprocessor Recommendations: Preprocessors are recommended to always assign the type 'part' as a common distinction between parts and documents. Further possible type classifications like "tool" or "raw material" will be described in future releases in separate sections. Other values such 'detail', 'assembly', or 'standard' may also be assigned, but should be related as sub-category to the primary 'part' using the product_category_relationship.

Postprocessor Recommendations: Postprocessors should recognize 'part' but also values of 'detail', 'assembly', and 'standard' as indicating the categorized product is a part.

Related Entities: There are no specific related entities.

1.1.3.2 product_category_relationship

The product_category_relationship entity establishes a category-subcategory relationship between two product_category entities.

Attributes
- The ***name*** attribute gives the word or group of words by which the relationship is referred.
- The ***description*** attribute provides additional descriptive information related about the relationship.
- The ***category*** attribute identifies the *super* category (more general) in the relationship.
- The ***sub_category*** attribute identifies the *sub* category (more specific) in the relationship.

ENTITY product_category_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	
category	type: entity = product_related_product_category	
sub_category	type: entity = product_related_product_category	

Preprocessor Recommendations: There is no standard mapping defined for the name attribute. The description attribute is optional, and may contain any appropriate or mutually agreed upon string.

Postprocessor Recommendations: Postprocessors should expect the category value to be a product_related_product_category with name 'part'. The value for the attribute sub-category will be a product_related_product_category with name 'detail', 'assembly', or 'standard' to further qualify the type of part.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#160 = PRODUCT('H24-1123.1', 'Fixture RX25B', '', (#20));
#170 = PRODUCT('DIN 932', 'Screw M3x15', '', (#20));
#250 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#10, #160, #170));
#260 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #250, #270);
#270 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#160));
```

Example 4: exchange file for part master with type classification

The following section combines the above discussed concepts and segments in a complete example.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the complete requirements of part master identification with context and part type classification are illustrated in Diagram 4. This figure also illustrates the recommended assignment of an organization to the product in the role 'id owner' to identify the scope (context of interpretation) for uniqueness and validity of the product id. The corresponding STEP exchange file encoding for the complete part master is illustrated in Example 4.
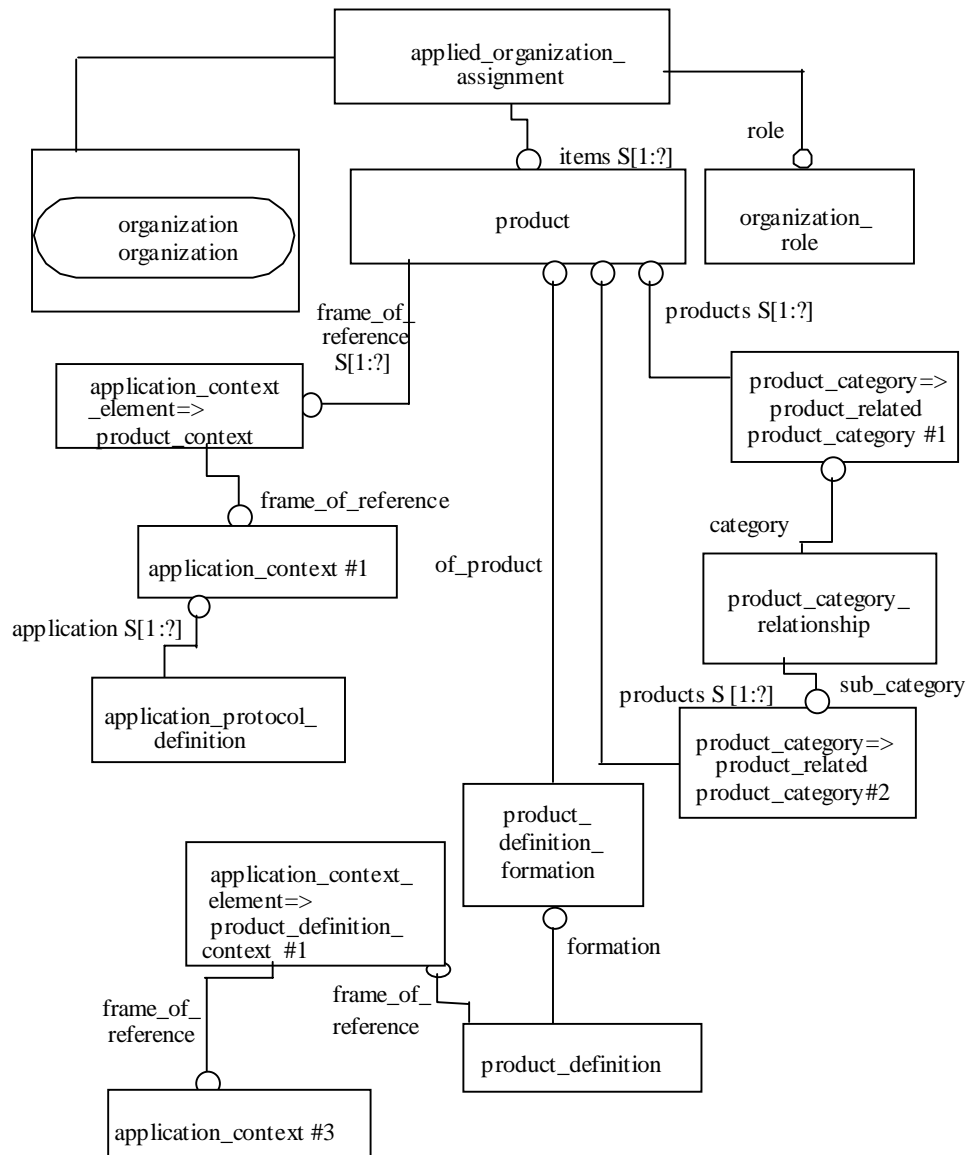
Diagram 4 : Complete Part Master with Context and Type Classification Instance Diagram

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('test', 'file'), '2;1');
FILE_NAME('pid2_p21a.stp', '1999-05-03T21:03:29+00:00', ('N.N.'), (''),
'', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}));
ENDSEC;
DATA;

/* part #1 */
#10 = PRODUCT('K01-42051', 'Bicycle Bell RX 3', $, (#20));
```

```
/* part context */
#20 = PRODUCT_CONTEXT('', #30, '');
#30 = APPLICATION_CONTEXT('');
#40 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema',
1998, #30);

/* part versions for part #1 */
#50 = PRODUCT_DEFINITION_FORMATION('02', 'lever modified', #10);
#60 = PRODUCT_DEFINITION_FORMATION('03', 'upper housing modified',
#10);
#70 = PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('', 'sequence', $, #50,
#60);

/* definition of view on version 03 of part #1 */
/* primary life_cycle_stage = design, primary application_domain =
mechanical design */
#80 = PRODUCT_DEFINITION('/NULL', $, #60, #90);
#90 = PRODUCT_DEFINITION_CONTEXT('part definition', #100, 'design');
#100 = APPLICATION_CONTEXT('mechanical design');

/* association of the id owner for part #1 */
#130 = APPLIED_ORGANIZATION_ASSIGNMENT(#140, #150, (#10, #160, #170));
#150 = ORGANIZATION_ROLE('id owner');

/* information on person and organization */
#140 = ORGANIZATION('ABC27166', 'Onyx AG', 'location');
#540 = PERSON('sarah.rijker@onyx.com', 'Rijker', 'Sarah', $, $, $);
#550 = PERSON_AND_ORGANIZATION(#540, #140);

/* part #2 and part #3 */
#160 = PRODUCT('H24-1123.1', 'Fixture RX25B', '', (#20));
#170 = PRODUCT('DIN 932', 'Screw M3x15', '', (#20));

/* part versions for part #2 and part #3 */
#180 = PRODUCT_DEFINITION_FORMATION('B', 'larger screw holes', #160);
#190 = PRODUCT_DEFINITION_FORMATION('15', '', #170);

/* view definition for version of part #2 */
#200 = PRODUCT_DEFINITION('/NULL', $, #180, #210);
#210 = PRODUCT_DEFINITION_CONTEXT('part definition', #215, 'design');
#215 = APPLICATION_CONTEXT('mechanical design');

/* view definition for version of part #3 */
#220 = PRODUCT_DEFINITION('/NULL', $, #190, #230);
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #240, 'design');
#240 = APPLICATION_CONTEXT('mechanical design');

/* part discriminator for parts #1 - #3 */
#250 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#10, #160, #170));
#260 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #250, #270);
#270 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#160));
#280 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #250, #290);
#290 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#10));
#300 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #250, #310);
#310 = PRODUCT_RELATED_PRODUCT_CATEGORY('standard', $, (#170));
```

```
ENDSEC;
END-ISO-10303-21;
```

Example 5: exchange file for complete part master with context and type classification

1.2   Known issues

1.2.1   application context information for product definition

The recommended mapping of the primary context perspective onto the product_definition_context referenced by product_definition.frame_of_reference conflicts with the current mapping of AP214. AP214 uses that context exclusively to assign product definition type information and requires the context defining the point of view on the product data to be captured in an additional product_definition_context that is attached via a product_definition_context_association. The usage guide chose an approach differing from AP214 DIS to ensure interoperability within the considered APs. On the other hand, this approach was chosen to allow a distinction between primary context and other contexts - a requirement that is currently not supported by 214 DIS. A ballot comment against AP214 DIS has been submitted.

2   Specific Part Type Classification

A simple basic type of classification of products in STEP works by assigning categories to product data items. These categories are identified by name labels that define the related classification. This type of classification is referred to as specific classification.

NOTE - As an advanced requirement there might be the need to classify product data items according to a classification system with explicit reference to the classification criteria and related properties of the product data items. This classification mechanism is called general classification. The PDM Schema currently only supports the specific classification of product data items via assigned categories, which are defined by labeling them with a name.

The PDM Schema supports the specific classification of parts and managed documents. The specific part type classification works in addition to the basic part vs. document type classification mechanism.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of specific part type classification are illustrated in Diagram 5.
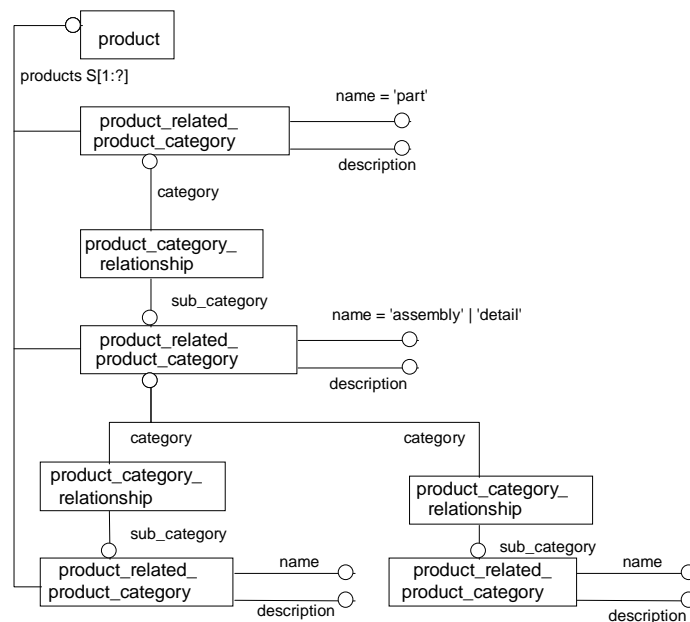
Diagram 5 : Specific Part Type Classification Instance Diagram

2.1.1.1   product_related_product_category

The product_related_product_category entity is a subtype of product_category.

Attributes
- The *name* attribute identifies the category via a label.
- The *description* provides additional characterization for the product category.
- The *products* attribute associates the category with the product entity instances to which it applies.

ENTITY product_related_ product_category	Attribute Population	Remarks
name	type: label = text	
description	type: text = string	OPTIONAL
products	type: entity = product	SET[1:?]

Preprocessor Recommendations: For reasons of robustness (see postprocessor recommendations below) preprocessors should list all products in the attribute product_related_product_category.items that are either direct members of that category or which belong to sub-categories of this category.
A given product category shall only be instantiated once in an exchange file.

It is recommended that all product_related_product_category instances used for the purposes of classification be interrelated in a single taxonomy. The root node of this taxonomy relates to the basic classification used to distinguish parts from documents. More specific categorisations are related to this basic category as sub-categories.

Since the specific classification capability relies exclusively on the name labels for the interpretation of the category semantics, exchanges of such information are only meaningful when the category names are agreed upon between the exchange partners. For the PDM Schema a basic hierarchy of categories for specific part classification is recommended (see Table 1). Exchange partners might extend that list based on bilateral agreements.

1st level category	2nd level category	3rd level category	...
part	detail	spare part	
	assembly	prototype part	
	standard		

Table 1 : Basic Part Classification Hierarchy

Postprocessor Recommendations: Postprocessors should also tolerate instances of product_category as opposed to product_related_product_category in the intermediate levels of a product category hierarchy definition. Processors should at least be robust enough to ignore the product_category instances. If these instances are ignored then the detailed information of the product category hierarchy is not completely transferred, nevertheless the assignment of products to basic categories is still maintained since these assignments always use product_related_product_categories.

Related Entities: The entity product_category is a supertype of product_related_product_category that does not have the attribute products. Thus a product_category can only be used as an intermediate node in a classification hierarchy to which no products are directly assigned.

2.1.1.2 product_category_relationship

The product_category_relationship entity establishes a category-subcategory relationship between two product_category entities.

Attributes
- The *name* attribute gives the word or group of words by which the relationship is referred.
- The *description* attribute provides additional descriptive information related about the relationship.
- The *category* attribute identifies the *super* category (more general) in the relationship.
- The *sub_category* attribute identifies the *sub* category (more specific) in the relationship.

ENTITY product_category_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	

category	type: entity = product_related_product_category	
sub_category	type: entity = product_related_product_category	

Preprocessor Recommendations: The PDM Schema does not have restrictions on potential nestings of category hierarchies besides that it is strongly recommended that the structure is acyclic. Nevertheless preprocessing systems should be aware that not all postprocessing systems might support an arbitrary deep structuring of hierarchy levels.

Postprocessor Recommendations: Postprocessors should at least support a two level category hierarchy. This is in alignment with the requirement to process the basic product instance type classification, which is to be done on two levels for product entity instances that model parts being assemblies or details.

Related Entities: Instances of product_category_relationship relate instances of product_category and product_related_product_category.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''), '2;1');
FILE_NAME('', '10.09.1999 14:18:28', ('n.n.'), (''), '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
#10 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #20, #30);
#20 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#40));
#30 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#40));
#40 = PRODUCT('comp_part', 'a_part', '', (#60));
#50 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #30, #100);
#60 = PRODUCT_CONTEXT('', #70, '');
#70 = APPLICATION_CONTEXT('');
#80 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema',
1998, #70);
#90 = PRODUCT_DEFINITION_FORMATION('1', '', #40);
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('spare part', $, (#40));
#110 = PRODUCT_CATEGORY_RELATIONSHIP('',$, #100, #120);
#120 = PRODUCT_RELATED_PRODUCT_CATEGORY('independent product', $,
(#40));
ENDSEC;
END-ISO-10303-21;
```

Example 6 : exchange file segment for specific part type classification

2.2 Known issues

2.2.1 Availability of specific part classification capability in AP 214 DIS

AP214 DIS does not support arbitrary levels of categories in the specific part classification capability.

3   Part Properties

The PDM Schema 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 the PDM Schema. A number of pre-defined property type names are also proposed for use when appropriate.

A special case of part properties is that of the part shape property - a representation of the geometrical shape model of the part. Various relationships are defined between external geometrical models to represent geometric model structure and the associated transformation information required for digital mock-up of assembly structures.

3.1   General Part Properties

The PDM Schema allows specifying properties associated with product data by linking a representation of the property values to the object with which the property is associated.

The PDM Schema optionally allows property identification that is independent from the actual association of a property to product data. In this case, the property definitions of a given type are additionally associated to a single 'general' identification for the particular type of property. The general_property entity collects multiple property_definitions that are of the same type, which may be associated to different elements of product data.

3.1.1   Properties Associated with Product Data

The PDM Schema allows specifying properties associated with product data. To specify properties associated with product data, a property definition tree is instantiated that links a representation of the property values to the described object that is the object with which the property is associated. The type of the property is given by the value of the attribute property_definition.name.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of property definitions associated with product data are illustrated in Diagram 6.
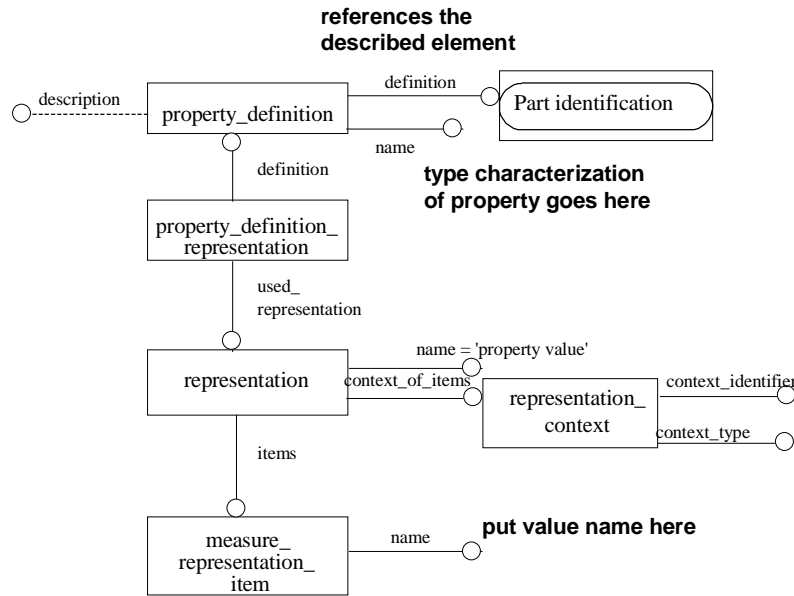
Diagram 6 : Property Definition Associated with Product Data Instance Diagram

3.1.1.1 property_definition

A property_definition is in the given context a property that characterizes a part. In the given context it applies to a product_definition that represents a life cycle and application domain specific view on a part.

ENTITY property_definition	Attribute Population	Remarks
name	type: identifier = string	Provides the property type name
description	type: text = string	OPTIONAL
definition	type: entity = here: product_definition	References product_definition as the element of part identification that is described by the property.

Preprocessor Recommendations: The name attribute characterizes the kind/type of the property. For the property_definition.name attribute, a number of predefined names for specific properties exist. Preprocessor should use these values where applicable.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.1.1.2 property_definition_representation

A property_definition_representation is in the given context an association between a property and its representation.

ENTITY property_definition_representation	Attribute Population	Remarks
definition	type: entity = property_definition	References associated property_definition
used_representation	type: entity = representation	References associated representation

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.1.1.3 representation

A representation in the context of part properties is a collection of one or more descriptive_representations_items related to a property_definition via a property_definition_representation.
.

ENTITY representation	Attribute Population	Remarks
name	type: label = STRING	shall be instaniated with 'property value'
items	type: entity = descriptive_representation_-item	the items that constitute the representation of the document property
context_of_items	type: entity = representation_context	is required to point to a representation_context with representation_context.type set to 'document parameters'

Preprocessor Recommendations: The name attribute shall be instantiated as 'property value' to indicate that the representation relates to a property value.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.1.1.4 representation_context

The representation_context defines the context of interpretation for the values of items in a representation.

Attributes
- The *context_identifier* attribute identifies the context.
- The *context_type* attribute specifies the type of the context.

ENTITY representation_context	Attribute Population	Remarks
context_identifier	type: label = string	
context_type	type: text = string	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.1.1.5 measure_representation_item

A measure_representation_item is in the property context a value element that participates as an item in one or more representations to define the respective properties. The given numeric value is associated with the relevant unit.

ENTITY measure_representation_item	Attribute Population	Remarks
name	type: label = STRING	the name attribute indicates the name of the represented property
value_component	type: select = measure_value	
unit_component	type: entity = unit	

Preprocessor Recommendations: For the unit_component it is recommend to do complex instantiations of named_unit with si_unit or named_unit with conversion_based_unit

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Entities #5020 to #5070 assert that the #290 defined view */
/* of part has a mass of 3KG                                 */
#5020 = PROPERTY_DEFINITION('mass property', $, #290);
#5030 = PROPERTY_DEFINITION_REPRESENTATION(#5020, #5040);
#5040 = REPRESENTATION('property value', (#5060), #5050);
#5050 = REPRESENTATION_CONTEXT('', '');
#5060 = MEASURE_REPRESENTATION_ITEM('mass', MASS_MEASURE(3000), #5070);
#5070 =(NAMED_UNIT(*) MASS_UNIT() SI_UNIT($,.GRAM.));
```

Example 7 : exchange file segment for property definition associated with product data

3.1.2 Independent Property Identification

The PDM Schema also optionally allows property identification that is independent from the actual association of the property to product data. In this case, the property definitions of a given type are additionally associated to a single 'general' identification for the particular type of property.

The general_property entity collects multiple property_definitions of the same type, by usage of the general_property_association entity. Multiple property definitions of the same type may exist and be associated to different elements of product data. The relationship between a general_property identification and a related property also indicates if the related property is 'definitional' (i.e. can be used to discriminate a given element against others) or 'non-definitional'.

The PDM Schema also supports the specification of relationships between two general property objects. These relationships may be used to indicate that the value of one property can be derived from the value of another property.   The relationship is modeled using the entity general_property_relationship.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of independent property identification are illustrated in Diagram 7.
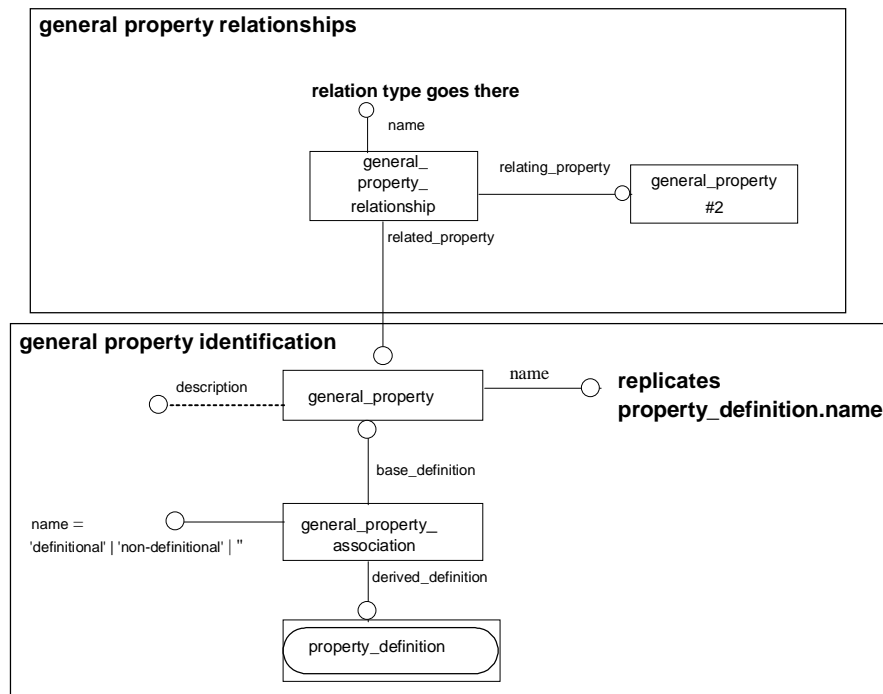
Diagram 7 : Independent General Property Identification and Relationship Instance Diagram

3.1.2.1 General_property

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

Attributes
- The *id* attribute provides an identification of a general property.
- The *name* attribute stores the type of the described property.
- The *description* attribute can be used to provide a further description of the property type.
-

ENTITY general_property	Attribute Population	Remarks
Id	Type: identifier = string	Recommended to be instantiated as "
Name	Type: label = string	Characterizes the property
Description	Type: text = string	OPTIONAL, provides further description of the property

Preprocessor Recommendations: General_property.name shall be replicated from the name attribute from the associated property_definition(s).   For general_property.name a number of predefined names for specific properties exist. Preprocessor should use these values where applicable. The instantiation of a general_property in association to a property_definition is not mandatory.

Postprocessor Recommendations: None specific.

Related Entities: General_property is to be associated via general_property_association to the characterized property_definition.

3.1.2.2 general_property_association

A general_property_association associates a given property with a general_property in order to collect property definitions of the same type. The entity also carries the information whether a given property can be used to distinguish the described object from others.

Attributes
- The *name* attribute indicates whether the described property can be used to distinguish the described element from others of the same kind.
- The *description* attribute can be used to provide a further description of the property type.
- The *base_definition* attribute references the general_property associated with the property_definition.
- The *derived_definition* references the characterized property_definition.

ENTITY general_property_association	Attribute Population	Remarks
name	type: label = string	Specifies whether the associated property_value object may be used to distinguish the described element from others of the same kind. A value of 'definitional' indicates that the associated property_value distinguishes it from others. The permissive list for the name attribute is 'definitional', 'non-definitional' or ''.
description	type: text = string	OPTIONAL
base_definition	type: entity = general_property	
derived_definition	type: entity = property_definition	

Preprocessor Recommendations: The value of the name attribute may be used to indicate if the associated derived_definition is a defining and distinguishing property for the part definition that is described by the property.

Postprocessor Recommendations: None specific.

Related Entities: General_property_association relates a property_definition and a general_property.

3.1.2.3 general_property_relationship

A general_property_relationship asserts a relationship between two independently identified general property identifications. A typical example of the use of this entity is to describe the 'derived' relation type, where the relating_property is specified as being derived from the related_property.

Attributes
- The *name* attribute.is used to describe the type of relationship between the properties.
- The *description* attribute can be used to provide optional further textual description.
- The *relating_property* attribute references the first of the related property pair.
- The *related_property* attribute references the second of the related property pair.

ENTITY general_property_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	OPTIONAL
relating_property	type: entity = general_property	

related_property	type: entity = general_property	

Preprocessor Recommendations:. The meaning of the relating_property and related_property attributes is specified further by the relation type indicated with general_property_relationship.name.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Entity #5000 establishes a property of type 'mass' independently */
/* of any value assignment of that property to product data. */
#5000 = GENERAL_PROPERTY('', 'mass property', $);


/* Entities #5010 and #5080 serve to collect all value assignments */
/* of the property of type 'mass' together under the independent */
/* general'mass' property definition */
#5010 = GENERAL_PROPERTY_ASSOCIATION('non-definitional', '', #5000,
#5020);
#5080 = GENERAL_PROPERTY_ASSOCIATION('non-definitional', '', #5000,
#5090);


/* Entities #5020 to #5070 assert that the in #290 defined view */
/* of part2 has a mass of 3KG                                    */
#5020 = PROPERTY_DEFINITION('mass property', $, #290);
#5030 = PROPERTY_DEFINITION_REPRESENTATION(#5020, #5040);
#5040 = REPRESENTATION('property value', (#5060), #5050);
#5050 = REPRESENTATION_CONTEXT('', '');
#5060 = MEASURE_REPRESENTATION_ITEM('mass', MASS_MEASURE(3000), #5070);
#5070 =(NAMED_UNIT(*) MASS_UNIT() SI_UNIT($,.GRAM.));


/* Entities #6090 to #6060 assert that the in #390 defined view */
/* of part2 has a mass of 6KG                                    */
#6090 = PROPERTY_DEFINITION('mass property', $, #390);
#6030 = PROPERTY_DEFINITION_REPRESENTATION(#6090, #6040);
#6040 = REPRESENTATION('property value', (#6060), #6050);
#6050 = REPRESENTATION_CONTEXT('', '');
#6060 = MEASURE_REPRESENTATION_ITEM('mass', MASS_MEASURE(6000), #5070);
```

Example 8 : exchange file segment for independent property identification

3.1.3 Pre-Defined Properties

For the representation of part properties, a number of property types have been pre-defined. It is recommended to use these values for the property_definition.name attribute where applicable.

- **Recyclability property** - A recyclability property is information concerning the ability to reuse objects or components of objects after their primarily intended usage.
  For recyclability properties of parts property_definition.name = **'recyclability property'** shall be used
- **Mass property** - a mass property is a quantity of matter that an object consists of.
  For mass properties of parts property_definition.name = **'mass property'** shall be used
- **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. For quality properties of parts property_definition.name = **'quality property'** shall be used.

- **Cost property** - a cost property is a property that specifies costs. For cost properties of parts property_definition.name = **'cost property'** shall be used.
- **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.

3.1.3.1 descriptive_representation_item

A descriptive_representation_item is in the property context a textual element that participates in one or more representations to define the respective properties.

- The *name* attribute characterizes the information modeled with the descriptive_representation_item.
- The *description* attribute defines a textual value as an instantiation of the modeled property.

ENTITY descriptive_representation_item	Attribute Population	Remarks
name	type: label = STRING	
description	type: text = STRING	The description is the value associated with the representation item in textual form

Preprocessor Recommendations: None specific

Postprocessor Recommendations: None specific

Related Entities: The descriptive_representation_items for a given property are collected in a representation.

3.1.4 Additional Part Properties

Several implementations and pilot projects have identified the requirement to exchange additional properties beyond the pre-defined values given in 3.1.3. One example of such a user defined property is the notes/remarks related to a product. A remark is some text that while associated with a part does not identify or describe it. These general remarks associated with part meta data are mapped to descriptive_representation_items related to the product_definition via a property_definition. This instantiation structure captures the actual text of a note/remark as a remarks property in the STEP file. The property_definition.name = '**remarks property**' may be used to identify such a property. An alternative external representation of related notes/remarks as a 'document' (see 5.1) or a 'file' (see 7.1) makes reference to but does not capture the actual text.

3.2 External Part Shape

The PDM Schema generally represents part master data. This part 'meta' data does not typically represent the detailed geometry of the part shape - this is the scope of CAD systems. Rather, the externally defined geometry is identified as an external representation of the part shape that is related to the part master data.

Part geometry is identified in the PDM Schema as a representation of a property of a part definition. The external part shape represents an external model referenced to an external file that is associated to the PDM master file.

3.2.1 Geometric Shape Property

Part geometry is identified in the PDM Schema as a representation of a property of a part definition. As with general part properties, a representation of a property is identified and linked to the product definition by the entity property_definition. For the part shape property, property_definition is specialised to the subtype product_definition_shape.

The external part shape is represented by the entity shape_representation; a subtype of representation used in general part properties. Shape_representations model external models that are referenced in files associated to the PDM master file.

STEP Application Protocols define various subtypes of shape_representations with differing constraints on the allowable representation_items to explicitly represent the detailed geometric model. In the context of the PDM Schema, it is generally assumed that these subtypes of shape_representation are not used.

In the scope of the PDM Schema it is generally assumed that shape_representations are placeholders for externally defined geometry. This geometry may be defined in STEP format as well as in native CAD format. In general it is not recommended to instantiate dedicated geometric elements as items of the shape_representations for PDM Schema files. However, certain detailed elements of the shape are required to be able to place and relate the external geometric models together. Therefore, placement information is placed in the set of items of each shape_representation. Placement information is modeled using the entity axis_placement a subtype of geometric_representation_item.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of independent property identification are illustrated in Diagram 8.
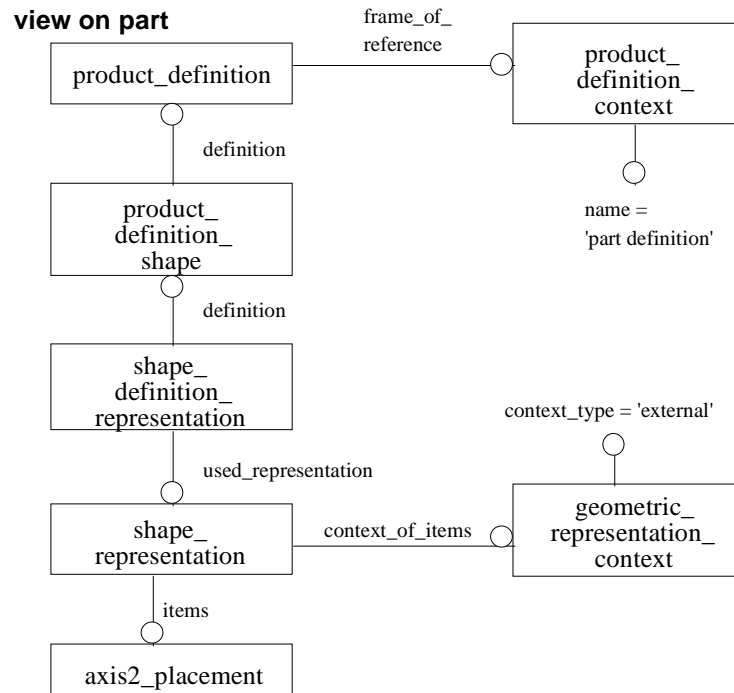
Diagram 8 : Assignment of Part Shape to the View of the Part Instance Diagram

3.2.1.1 product_definition_shape

Product_definition_shape represents the shape of a product. The shape may be a conceptual shape for which a specific geometric representation is not required.

ENTITY product_definition_shape	Attribute Population	Remarks
name	type: label = string	no standard mapping exists

		should be instantiated as empty string
description	type: text = string	OPTIONAL
definition	type: entity = product_definition	reference to an instance of product_definition to which the shape information is attached

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.2.1.2 shape_definition_representation

A shape_definition_representation links the definition of a shape, i.e. the product_definition_shape, with its representation, i.e. a shape_representation.   For a given product_definition_shape alternative representations might be given by multiple occurrences of shape_definition_representation that each link a representation to one product_definition_shape.

Attributes
- The *definition* attribute is used to point to the product_definition_shape for which the representation of shape is provided.
- The *used_representation* references an instance of shape_representation that provides a representation of the part shape.

ENTITY shape_definition_representation	Attribute Population	Remarks
definition	type: entity = product_definition_shape	
used_representation	type: entity = shape_representation	

Preprocessor Recommendations: Some APs allow the assignment of a role ('detailed representation' or 'idealized representation') to a shape_definition_representation by using *name_attribute*.named_item. However, it is recommended to use a document content property (see 9.2) to record this type of information instead of using *name_attribute*.attribute_value.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.2.1.3 shape_representation

Shape_representation is a subtype of representation that is specifically used to represent shape information.

Attributes
- The *name* attribute unambiguously identifies the geometric model within the CAD environment.
- The *items* attribute collects the items of the shape_representation.
- The *context_of_items* attribute references a geometric_representation_context that establishes the coordinate space for the shape_representation.

ENTITY shape_representation	Attribute Population	Remarks
name	type: label = string	

items	type: set of representation items	a set of *representation_item*s that are related in the context_of_items
context_of_items	type :entity = geometric_representation_context	reference to an instance of *geometric_representation_context* or an complex instance of (*geometric_representation-_context, global_uncertainty-_assigned_context*)

Preprocessor Recommendations: Some APs allow the assignment of a version to a shape_representation by using *version_assignment*.assigned_id with fixed value: *version_assignment*.role.name = 'version id'. It is recommended not to assign versions to shape_representations.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.2.1.4 geometric_representation_context

The geometric_representation_context defines the coordinate space in which the geometric model defined by a shape_representation resides.

Attributes
- The *context_identifier* attribute identifies the coordinate system.
- The *context_type* attribute specifies the type of context.
- The *coordinate_space_dimension* **attribute** defines the dimensionality of the coordinate space.

ENTITY geometric_representation_context	Attribute Population	Remarks
context_identifier	type: label = string	
context_type	type: text = string	should be instantiated with 'external'
coordinate_space_dimension	type: dimension_count = INTEGER	it is recommend to instantiate with 3

Preprocessor Recommendations: The explicit representation of geometry is not in the scope of the PDM Schema. The PDM Schema supports exclusively shape_representations that are externally defined. Thus the context_type attribute shall be instantiated with 'external'.

Postprocessor Recommendations: None specific.

Related Entities: geometric_representation_context specifies the coordinate system for the use by shape_representations.

3.2.1.5 geometric_representation_item

Geometric_representation_items are items for product data representation with the additional meaning of having geometric context. Geometric_representation_items are subtyped to a variety of entities in the context of geometric representations. The subtypes relevant for the PDM Schema are axis_placements that can be used to define transformations between shape_representations.

Attributes
- The *name* attribute is used to name the geometric_representation_item.

ENTITY geometric_representation_item	Attribute Population	Remarks
name	type: label = string	

Preprocessor Recommendations: only subtype of geometric_representation_item should be instantiated.

Postprocessor Recommendations: None specific.

Related Entities: most relevant in the scope of the PDM Schema are axis2_placement_3d to define transformation information.

3.2.1.6 axis2_placement_3d

Axis2_placement_3d is a geometric_representation_item that specifies the location and orientation in three-dimensional space of two mutually perpendicular axes.

Attributes
- The *name* attribute provides a name for the placement.
- The *location* attribute defines the spatial position of the reference point and origin of the associated placement coordinate system.
- The *axis* attribute defines the exact direction of the local Z axis.
- The *ref_direction* attribute can be used to determine the direction of the local X axis.

ENTITY axis2_placement_3d	Attribute Population	Remarks
name	type: label = string	
location	type: entity = cartesian_point	
axis	type: entity = direction	OPTIONAL
ref_direction	type: entity = direction	OPTIONAL

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: If necessary an adjustment to ref_direction has to be made to maintain orthogonality to the axis direction. If axis or ref_direction are omitted, these directions are taken from the geometric coordinate system.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#290 = PRODUCT_DEFINITION('p2v', 'view on part2', #280, #130);
/* Entities #600 to #660 define the geometric model for 'part2' */
#600 = PRODUCT_DEFINITION_SHAPE('shape_p2',$,#290);
#610 = SHAPE_DEFINITION_REPRESENTATION(#600,#620);
#620 = SHAPE_REPRESENTATION('sol2',(#670),#630);
#630 = GEOMETRIC_REPRESENTATION_CONTEXT('c2','external',3);

/* Entities #640 to #670 are the elements of shape representation */
#640 = CARTESIAN_POINT('', (3.0E+001, 0.0E+000, 1.0E+001));
#650 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#660 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#670 = AXIS2_PLACEMENT_3D('', #640, #650, #660);
```

Example 9 : exchange file segment to associate shape to part views

3.2.2 Portions of the Part Shape

The PDM Schema supports the requirement to identify explicitly a portion of shape that has to be associated with other information – including dedicated geometric models. Property information can be linked to shape aspects via property definitions. A prominent special case is the assignment of dedicated shape_representations as shown in the diagram below.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of independent property identification are illustrated in   Diagram 9.
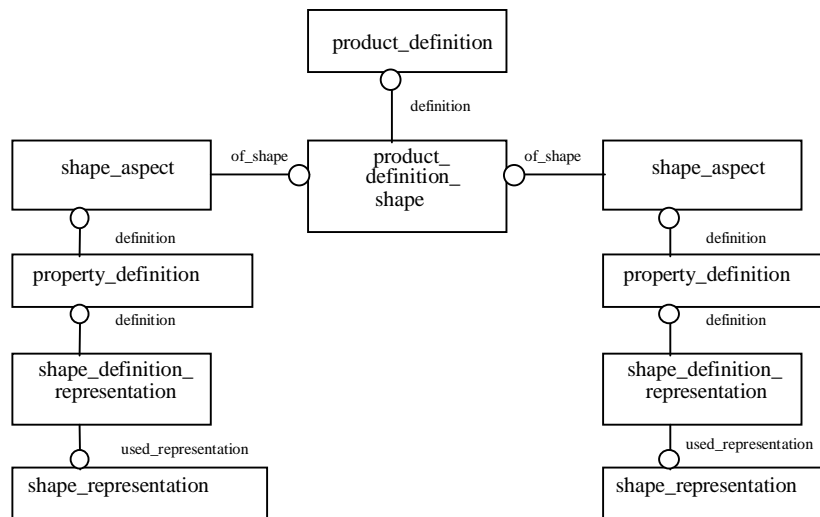
Diagram 9 : Identification and Representation of Portions of Part Shape Instance Diagram

3.2.2.1 shape_aspect

Attributes
- The *name* is the organizational name of the shape_aspect.
- The *description* attribute provides an optional textual description of the shape_aspect.
- The *of_shape* attribute references the product_definition_shape of which the shape_aspect is a portion.
- The *product_definitional* attribute indicates whether the portion of shape lies on the physical boundary of the part shape.

ENTITY shape_aspect	Attribute Population	Remarks
name	type: label = STRING	
description	type: text = STRING	OPTIONAL
of_shape	type: entity = product_definition_shape	
product_definitional	type: LOGICAL	If a value of TRUE is given,

		then it is asserted that the shape_aspect is on the physical boundary of the product_definition_shape.

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#700 = PRODUCT_DEFINITION_SHAPE('shape_p',$,#260);
#710 = SHAPE_ASPECT('aspect1',$,#700,.T.);
/* Entities #730 to #790 define the geometry related to  */
/* shape_aspect #710                                      */
#730 = PROPERTY_DEFINITION('shape for aspect1',$,#710);
#740 = SHAPE_DEFINITION_REPRESENTATION(#730,#750);
#750 = SHAPE_REPRESENTATION('sa1',(#795),#760);
```

Example 10: exchange file segment for identification of shape portions

3.2.3 Relating Externally Defined Part Shape to an External File

The PDM Schema generally represents part master data. This part 'meta' data does not typically represent the detailed geometry of the part shape - this is the scope of CAD systems. Rather, the externally defined geometry is identified as an external representation of the part shape that is related to the part master data. Part geometry is identified in the PDM Schema as a representation of a property of a part definition - it is actually in externally referenced files. These files are often generated by CAD systems, and contain the detailed geometric shape representation with dedicated geometric elements.

The external part shape is an external geometric model related to the part master data and is referenced as an external CAD file. Specifics of external file identification are described in Section 6. The external file that contains the detailed geometry is attached to the part master data in two ways:
- Related to the product identification data using applied_document_reference.
- Related to the shape_representation representing the external geometric model using the entities property_definition and property_definition_representation.

It is recommended that applied_document_reference be used in general to relate an external file representing the CAD model to the product_definition of a part identification (see Section 10). If the external file representing the CAD model exists alone as an unmanaged external file reference, then the document_reference should be applied directly from the document_file (see Section 10.2). If the external file representing the CAD model exists as a constituent file of a managed document (using the 'Document as Product' approach), the document_reference should be applied from the managed document via the document_product_equivalence (see Section 10.1).

If the CAD file contains the definitional shape of the part, then an external geometric model should be represented (using shape_representation) and the external CAD file should also be related to this shape_representation using property_definition and property_definition_representation.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of relating part shape to the external CAD file is illustrated in Diagram 10.
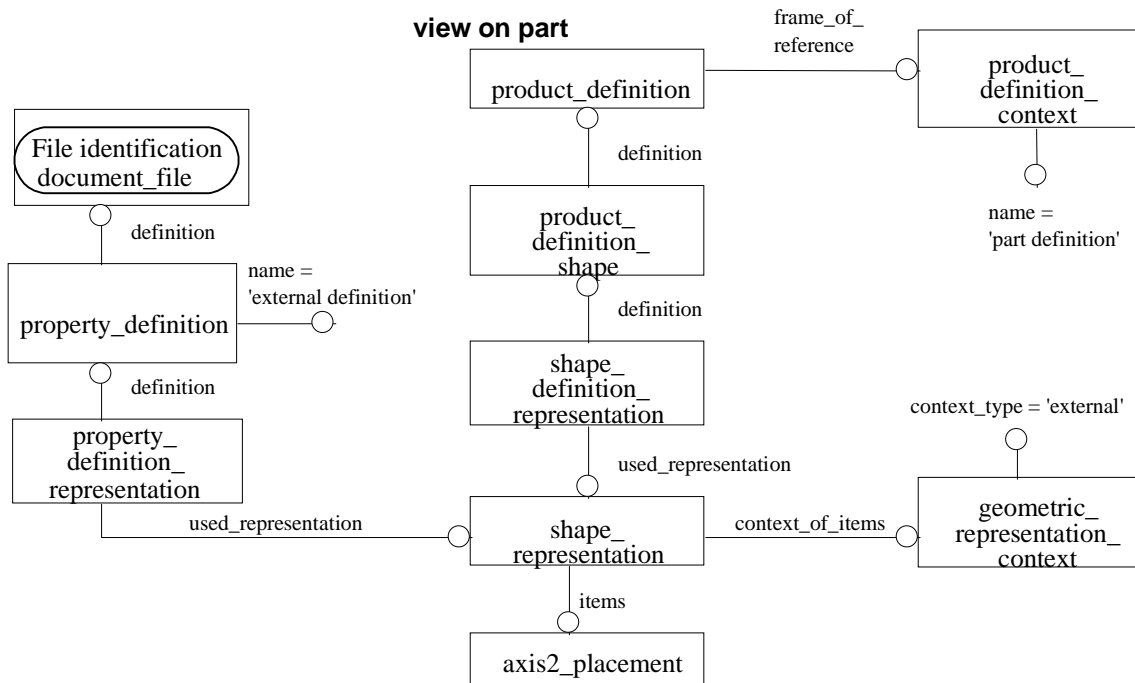
Diagram 10: Part Shape Related to External File Instance Diagram

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Entities #370 to #497 define a managed document which */
/* contains a representation of the shape of 'part2'     */
#370 = PRODUCT('p2_shape_doc', 'document reflecting
         shape of part2', '', (#20));
#380 = PRODUCT_RELATED_PRODUCT_CATEGORY('document', '', (#370));
#390 = DOCUMENT_TYPE('configuration controlled document version');
#400 = DOCUMENT('', '', '', #390);
#410 = APPLIED_DOCUMENT_REFERENCE(#290, 'equivalence', (#400));
#420 = OBJECT_ROLE('mandatory', '');
#430 = ROLE_ASSOCIATION(#420, #410);
#440 = PRODUCT_DEFINITION_FORMATION('1', '', #370);
#450 = DOCUMENT_PRODUCT_EQUIVALENCE('equivalence', '', #400, #440);
#460 = PRODUCT_DEFINITION_CONTEXT('digital document definition', #10,
'';
#470 = PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS('p2_shape_d',
   'digital document for part2 shape', #440, #460,(#475));
```

```
#475 = DOCUMENT_FILE('p2_shape_file', '', '', #485, ' ', ' ');
#480 = DOCUMENT_REPRESENTATION_TYPE('digital',#475);
#485 = DOCUMENT_TYPE('');
#490 = IDENTIFICATION_ROLE('document_source',$);
#495 = APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT('',#490,#497,(#475));
#497 = EXTERNAL_SOURCE('part1_geometry.stp');

/* Entities #600 to #660 define the geometric model for 'part2' */
#600 = PRODUCT_DEFINITION_SHAPE('shape_p2',$,#290);
#610 = SHAPE_DEFINITION_REPRESENTATION(#600,#620);
#620 = SHAPE_REPRESENTATION('sol2',(#670),#630);
#630 = GEOMETRIC_REPRESENTATION_CONTEXT('c2','external',3);
#640 = CARTESIAN_POINT('', (3.0E+001, 0.0E+000, 1.0E+001));
#650 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#660 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#670 = AXIS2_PLACEMENT_3D('', #640, #650, #660);

/* Entities #680 and #690 indicate that the shape_representation #620 /
/* is externally defined by the file represented by #475          */
#680 = PROPERTY_DEFINITION('external definition',$,#475);
#690 = PROPERTY_DEFINITION_REPRESENTATION(#680,#620);
```

Example 11: exchange file segment for externally defined geometry in managed documents

```
/* Entities #500 to #560 define the geometric model for 'part1' */
#500 = PRODUCT_DEFINITION_SHAPE('shape_p1',$,#230);
#510 = SHAPE_DEFINITION_REPRESENTATION(#500,#520);
#520 = SHAPE_REPRESENTATION('sol1',(#570),#530);
#530 = GEOMETRIC_REPRESENTATION_CONTEXT('c1','external',3);
#540 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 1.0E+001));
#550 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#560 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#570 = AXIS2_PLACEMENT_3D('', #540, #550, #560);

/* Entities #575 to #585 state that the shape_representation #520 */
/* is externally defined in a digital file with the location */
/* 'part2_geometry.stp'                                     */
#575 = APPLIED_DOCUMENT_REFERENCE(#230,(#576));
#576 = DOCUMENT_FILE('p1_shape','',$,#577,'','');
#577 = DOCUMENT_TYPE('geometry');
#578 = ROLE_ASSOCIATION(#579,#575);
#579 = OBJECT_ROLE('mandatory',$);
#580 = DOCUMENT_REPRESENTATION_TYPE('digital',#576);
#581 = IDENTIFICATION_ROLE('document_source',$);
#582 = APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT('',#581,#583,(#576));
#583 = EXTERNAL_SOURCE('part2_geometry.stp');
#584 = PROPERTY_DEFINITION('external definition',$,#576);
#585 = PROPERTY_DEFINITION_REPRESENTATION(#584,#520);
```

Example 12: exchange file segment for externally defined geometry in flat files

3.3   *External Geometric Model Structure*

External geometric models may be related together to form hierarchical geometric model structures. The orientation and location of two shape_representations relative to each other can be defined using a subtype of shape_representation_relationship that references a transformation. For example, one might want to

specify such a transformation to geometrically relate representations of the part shape in order to relate the shape of components in an assembly structure for digital mock-up applications.

3.3.1 Relating Part Shape

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of relating shape representations is illustrated in Diagram 11.
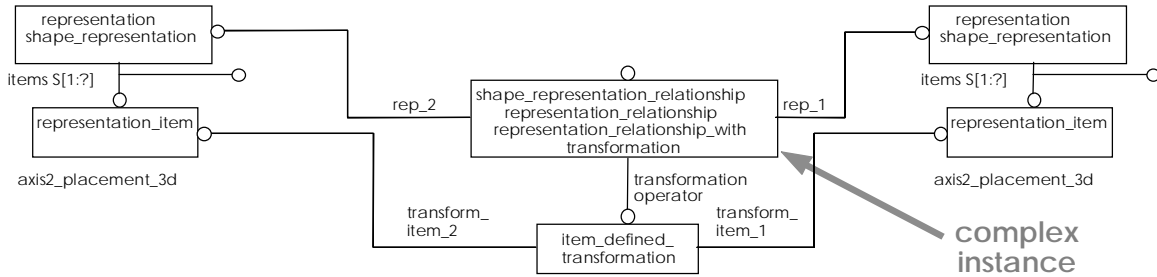

Diagram 11: Relating Shape Representations with Transformation Instance Diagram (item_defined_transformation example)

3.3.1.1 shape_representation_relationship

The entity shape_representation_relationship_is a subtype of representation_relationship. The subtype adds specific local constraints that ensures that it defines a relationship between two shape_representations. To define a relationship between two shape_representations that is established via a transformation, a complex instantiation of shape_representation_relationship AND representation_relationship_with_transformation is used.

Attributes
- The *name* attribute is used to name the relationship.
- The *description* attribute can be used to provide optional further textual description.
- The *rep_1* attribute references the first of the related shape_representations.
- The *rep_2* attribute references the second of the related shape_representations.

ENTITY shape_representation_relationship	Attribute Population	Remarks
Name	type: label = string	
Description	type: text = string	OPTIONAL
rep_1	type: entity = shape_representation	
rep_2	type: entity = shape_representation	

Preprocessor Recommendations: The meaning of the rep_1 and rep_2 attributes is specified further by the relation type indicated by the attribute shape_representation_relationship.name.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.3.1.2 representation_relationship_with_transformation

The entity representation_relationship_with_transformation_is a subtype of representation_relationship. The subytype adds the attribute transformation_operator as a reference to a transformation. To define a relationship between two shape_representations that is established via a transformation, a complex instantiation of shape_representation_relationship AND representation_relationship_with_transformation is used.

Attributes
- The *name* attribute is used to name the relationship.
- The *description* attribute can be used to provide optional further textual description.
- The *rep_1* attribute references the first of the related shape_representations.
- The *rep_2* attribute references the second of the related shape_representations.
- The *transformation_operator* attribute specifies the transformation operator that defines the relationship.

ENTITY shape_representation_relationship	Attribute Population	Remarks
Name	See shape_representation_relationship	
Description	See shape_representation_relationship	OPTIONAL
rep_1	See shape_representation_relationship	
rep_2	See shape_representation_relationship	
Transformation_operator	type: entity item_defined_transformation or functionally_defined_transformation	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#520 = SHAPE_REPRESENTATION('sol1',(#570),#530);
#530 = GEOMETRIC_REPRESENTATION_CONTEXT('c1','external',3);
#540 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 1.0E+001));
#550 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#560 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#570 = AXIS2_PLACEMENT_3D('', #540, #550, #560);
#1070 = SHAPE_REPRESENTATION('s_ass',(#1110,#1140),#1080);
#1080 = GEOMETRIC_REPRESENTATION_CONTEXT('ca','',3);
#1110 = AXIS2_PLACEMENT_3D('', #1090, #1095, #1100);
#1120 = CARTESIAN_POINT('', (1.0E+001, 1.0E+000, 1.0E+001));
#1125 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#1130 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#1220 =(REPRESENTATION_RELATIONSHIP('','',#520,#1070)
   REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#1230)
   SHAPE_REPRESENTATION_RELATIONSHIP());
#1230 =ITEM_DEFINED_TRANSFORMATION('p1t','',#1110,#570);
```

Example 13: exchange file segment for relating shape representations

3.3.2 Relating portions of shape

By using the above described shape_representation_relationship concept, geometric relations of individual portions of shape defined as shape_aspects can be represented.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of relating portions of shape are illustrated in Diagram 12.
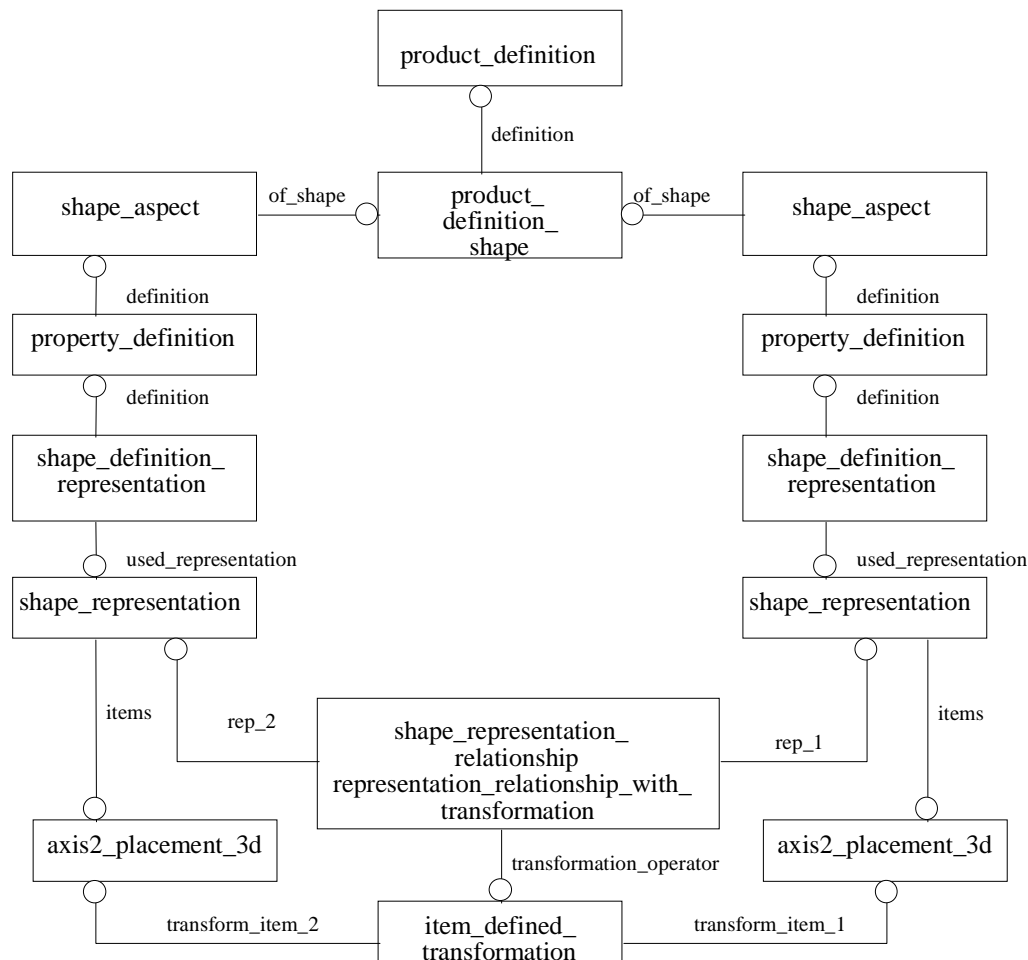
Diagram 12: Specifying Geometric Relations between Portions of Shape Instance Diagram

3.4   Relative Orientation and Location of Related Geometric Models

The relative orientation and location of two shape_representations to each other can be defined with the shape_representation_relationship that reference transformations. One might want to specify such transformations to geometrically relate representations of portions of the part shape to relate the shape of components in an assembly structure in order to define the geometric relations.

The PDM Schema supports two methods to define geometric relationships between external models:

- *implicitly* by identifying two items that establish a reference relationship via an item_defined_transformation,

- *explicitly* with the help of a functionally_defined_transformation. It is generally recommended that cartesian_transformation_operator be used to define a functionally_defined_transformation.

3.4.1 Implicitly defined transformations between geometric models

A relationship can be defined implicitly by referencing an item_defined_transformation that has two reference points to specify origin and target of the transformation.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of implicit definition of transformation between the elements of two representations are illustrated in Diagram 13.
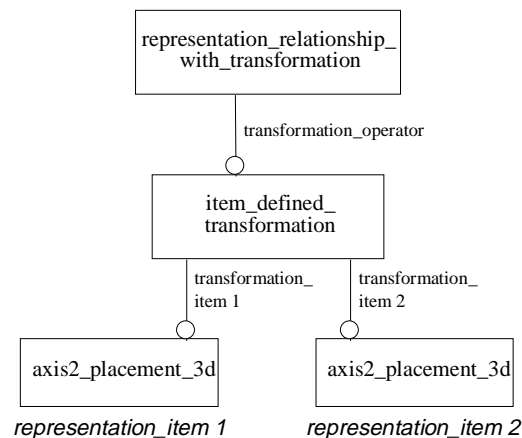
Diagram 13: Implicitly Defined Geometric Relations Instance Diagram

3.4.1.1 item_defined_transformation

An item_defined_transformation models a transformation performed by defining two representation_items before and after applying the transformation function The transformation function is not explicitly provided, but it is derived through its relationship to the representation_items.

Attributes
- The *name* attribute provides a name for the transformation.
- The *description* attribute can be used to provide optional further textual description.
- The *transform_item_1* references the first item (origin for the transformation).
- The *transform_item_2* attribute references the second item (target for the transformation).

ENTITY item_defined_transformation	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	OPTIONAL
transform_item_1	type: entity = representation_item	here: axis2_placement_3d
transform_item_2	type: entity = representation_item	here: axis2_placement_3d

Preprocessor Recommendations: To be meaningful the item_defined_trasnformation with axis2_placement_3d requires that the representations that include transfrom_item_1 and transform_item_2 share the same geometric_representation_context.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

3.4.2 Explicitly defined transformations between geometric models

Relationships between geometric models can be made modeled in an explicit manner by using instances of cartesian_transformation_operator.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of explicit definition of transformation between the elements of two representations is illustrated in Diagram 14, Diagram 15, and Diagram 16.
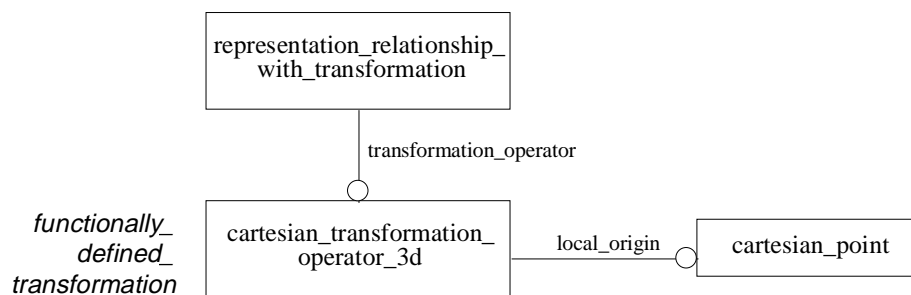
Diagram 14: Explicit Representation of a Translation Instance Diagram

The scale attribute with a positive value different from 1.0 allows uniform scaling in addition to translation or identity.
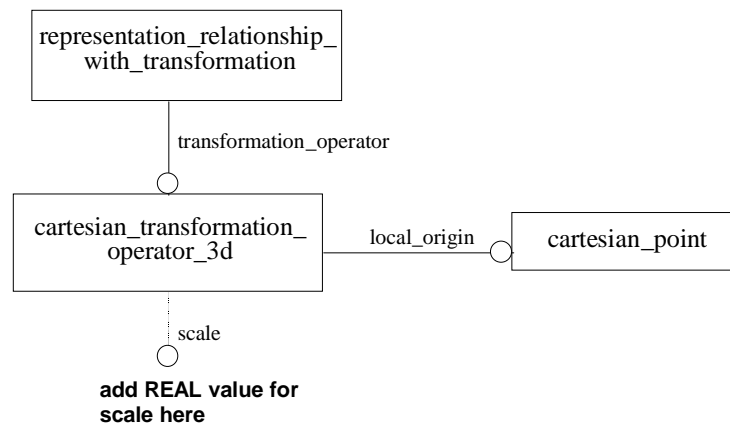
Diagram 15: Explicit Representation of a Translation with Scaling Instance Diagram

An additional rotation or mirroring requires the definition of the three **axis** attributes. If all optional attributes are set, a composition of all kinds of transformation is possible.
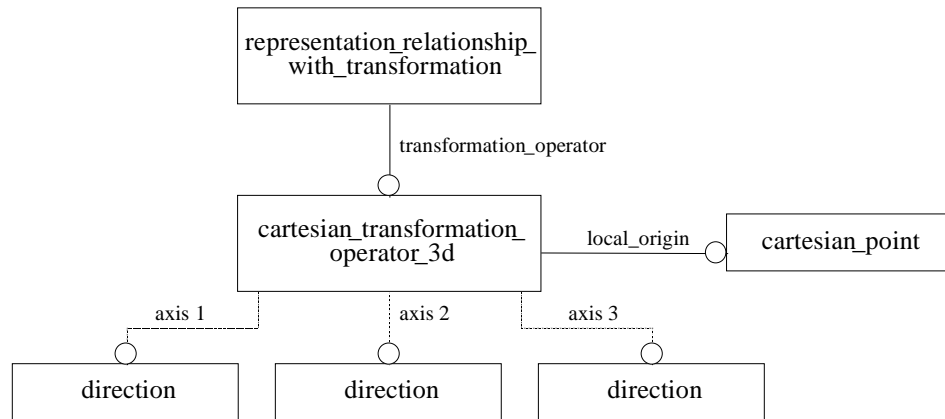
Diagram 16: Explicit Representation of a Translation with Rotation Instance Diagram

3.4.2.1 cartesian_transformation_operator

Cartesian_transformation_operator defines a geometric transformation composed of translation and rotation. Mirroring and scaling should not be used in the context of representing the geometric equivalent for an assembly structure. There are two subtypes of cartesian_transformation_operator:

- cartesian_transformation_operator_2d,
- cartesian_transformation_operator_3d.

Attributes
- The *name* attribute (inherited from supertype representation_item) provides a naming capability.
- The *name* attribute (inherited from functionally_defined_transformation) provides a naming capability.
- The *description* attribute can be used to provide optional further textual description.
- The *axis1* attribute is the X axis direction of the transformation target.
- The *axis2* attribute is the Y axis direction of the transformation target.
- The *local_origin* attribute is the required translation specified as a cartesian point. The actual translation included in the transformation is from the geometric origin to the local origin.
- The *scale* attribute is the scaling value specified for the translation.
- The *axis3* attribute is the Z axis direction of the transformation target.

ENTITY cartesian_transformation_operator	Attribute Population	Remarks
representation_item.name	type: label = string	
functionally_defined_transformation.name	type: label = string	
description	type: text = string	OPTIONAL
axis1	type: entity = direction	OPTIONAL
axis2	type: entity = direction	OPTIONAL
local_origin	type: entity = cartesian_point	
scale	type: REAL	OPTIONAL
axis3	type: entity direction	OPTIONAL Attribute specific to cartesian_transformation_operator_3d

Preprocessor Recommendations: A cartesian_transformation_operator is a subtype of geometric_representation_item, and has attributes populated with geometric entities. In consequence an instance of a cartesian_transformation_operator must be founded in (i.e. must exist as an element of) a shape_representation. It is thus recommended that the cartesian_transformation_operator be referenced as an item of the shape_representation that is the value of the attribute rep_2 of a shape_representation_relationship (e.g. the 'parent' shape_representation). In the context of an assembly, the rep_2 attribute points to the assembly definition as a target for the transformation of the component instance shape. This recommendation interprets the transformation to be part of the assembly shape rather than of the component shape.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

A cartesian_transformation_operator_3d defines a geometric transformation composed of translation, rotation, mirroring and uniform scaling. Depending on what kind of transformation is implemented, certain optional attributes are required. With only local_origin given as an attribute, the transformation may be an identity or a translation.

3.4.3 Conversion from Implicit to Explicit Transformation Information

In STEP, there are two possibilities of representing a transformation between two geometric models. In an implicitly defined transformation the item_defined_transformation entity is used with two representation_items (mostly axis2_placement_3ds) before and after applying the transforming function. In an explicitly defined transformation, this function is given by an instance of cartesian_transformation_operator. The operations allowed in this context are rotation and translation.

The intention of this paragraph is to demonstrate how to extract the transforming function from an item_defined_transformation and model to an explicitly defined transformation from this information. Therefore, an example is constructed. First, the implicit way is shown, then the conversion will be calculated and finally the explicit representation is described.

Implicitly defined transformation

To implicitly define a transformation between two representation_items, the following entities and attributes are needed (see Diagram 17)
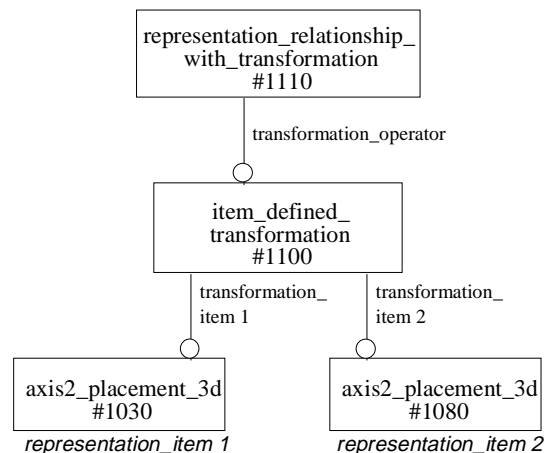
Diagram 17: Implicit Transformation Instance Diagram

This structure can be rediscovered in the following fragment from a Part 21 file:

```
/* representation_item 1 */

#1000=CARTESIAN_POINT('', 1.0, 1.0, 3.0);
#1010=DIRECTION('', 0.0, -0.8660254, 0.5);
#1020=DIRECTION('', 1.0, 0.0, 0.0);
#1030=AXIS2_PLACEMENT_3D('', #1000, #1010, #1020);

/* representation_item 2 */

#1050=CARTESIAN_POINT('', 2.0, 2.0, 1.0);
#1060=DIRECTION('', 0.0, 0.0, 1.0);
#1070=DIRECTION('', 0.7071068, 0.7071068, 0.0);
#1080=AXIS2_PLACEMENT_3D('', #1050, #1060, #1070);

/* transformation */

#1100=ITEM_DEFINED_TRANSFORMATION('', '', #1030, #1080);
#1110=REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION('', '', #800,
#900, #1100);
```

Example 14: file segment for implicitly defined transformation

Conversion

The first step is now to extract the two matrices implicitly given by each of the two placements. The axis2_placement_3d has a name, a location and two axes as attributes. The axes are the axis and ref_direction attribute, where axis is the placement Z axis direction and the ref_direction is an approximate to the placement X axis direction. From this information, a right-handed coordinate system is computed:

Let z be the placement Z axis direction and a be the approximate placement X axis direction. Approximate here means that a and z are not required to be orthogonal. Then the exact placement X axis direction is given as $x = \langle a - (a \cdot z)z \rangle$ and the placement Y axis direction calculates to $y = \langle z \times x \rangle$. For the first representation item, the following calculations would result:

$$\text{axis: } z = \begin{pmatrix} 0.0 \\ -0.8660254 \\ 0.5 \end{pmatrix}, \qquad \text{ref\_direction: } a = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \end{pmatrix},$$

therefore $x = \langle a - (a \cdot z)z \rangle = \langle a - 0z \rangle = a$ because z and a are already orthogonal in this example.

$$\text{Next step is calculating } y \text{ using the vector product: } y = \langle z \times x \rangle = \begin{pmatrix} 0.0 \\ 0.5 \\ 0.8660254 \end{pmatrix}.$$

So the geometric function which leads from the coordinates of the global coordinate system to those of the first axis placement is represented by the rotation matrix A given by the three vectors x, y and z plus the translation vector t given by the axis2_placement_3d's location attribute:

$$A = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & -0.8660254 \\ 0.0 & 0.8660254 & 0.5 \end{pmatrix}, \qquad t = \begin{pmatrix} 1.0 \\ 1.0 \\ 3.0 \end{pmatrix}$$

In the same way, the matrix B and the vector u are computed from the second axis placement:

$$B = \begin{pmatrix} 0.7071068 & -0.7071068 & 0.0 \\ 0.7071068 & 0.7071068 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}, \qquad u = \begin{pmatrix} 2.0 \\ 2.0 \\ 1.0 \end{pmatrix}$$

Geometrically, the matrix A defines a 60° rotation around the global X axis and the matrix B gives a 45° rotation around the global Z axis.

To get the explicit transformation from the information gained so far, the matrices have to be combined. The idea is as follows: To move a point from a location within the first placement into a location within the second one, three steps have to be made:

1. First, the point has to be multiplied with the inverted matrix A^{-1} to undo the rotation, which occurs when going from the global coordinate system into the first placement system.
2. Next, multiply it with the second matrix B to get it into the right position for the target placement.
3. Finally, a translation vector is needed to put the point into its correct location within the second axis placement. Calculation of this vector can be seen below.

As A is a rotation matrix, the inverted matrix $A^{-1} = A^{T}$, the transposed matrix. Steps 1 and 2 can be combined:

$$C = BA^{-1} = \begin{pmatrix} 0.7071068 & -0.3535534 & -0.6123724 \\ 0.7071068 & 0.3535534 & 0.6123724 \\ 0.0 & -0.8660254 & 0.5 \end{pmatrix}$$

The translation vector needed is

$$v = u - Ct = \begin{pmatrix} 3.4835639 \\ -0.8977775 \\ 0.3660254 \end{pmatrix}$$

This means moving any point P from a location within the first placement into the second one follows the calculation

$$P' = C \cdot P + v$$

Explicitly defined transformation

This information will now be used in a Part 21 file example, using an instance of the cartesian_transformation_operator_3d to perform a translation with rotation, where the columns of the matrix C are the three axis attributes and the vector v is the local_origin attribute. The structure of the file is shown in the following diagram:
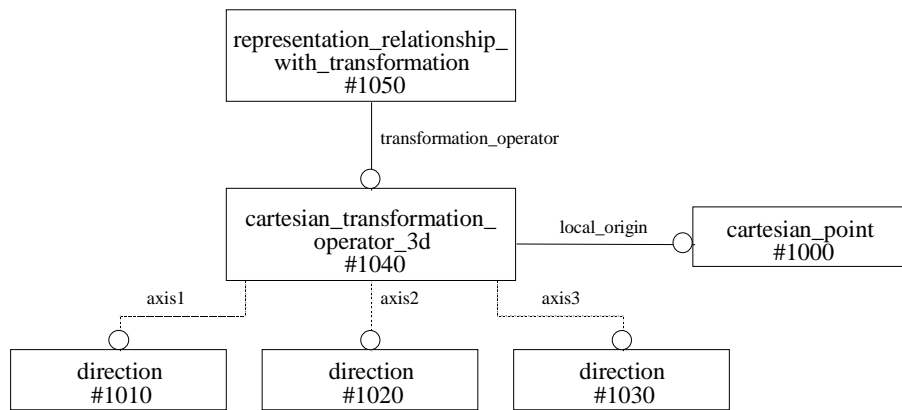
Diagram 18: Explicit Transformation Instance Diagram

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* translation vector v */

#1000=CARTESIAN_POINT('', 3.4835639, -0.8977775, 0.3660254);

/* rotation matrix C */

#1010=DIRECTION('', 0.7071067811865, 0.7071067811865, 0.0);
#1020=DIRECTION('', -0.3535533905933,  0,3535533905933,
0.8660254037844);
#1030=DIRECTION('', -0.6123724356958, 0.6123724356958, 0.5);

/* transformation */

#1040=CARTESIAN_TRANSFORMATION_OPERATOR_3D('', '', $, #1010, #1020,
#1000, $, #1030);
#1050=REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION('', '', #800,
#900, #1040);
```

Example 15: file segment for explicit transformation

3.5   Known issues

3.5.1   Material properties

Some application protocols such as AP214 use specific subtypes to specify material properties. These subtypes are currently not included in the PDM Schema.

3.5.2   Mapping of part properties in AP 214

The requirement to represent properties for parts is in AP214 always mapped onto an AIM structure that extends the property_definition tree with a general_property. In consequence AP214 compliant implementations are forced to always instantiate general_property when part properties shall be represented. We feel that this is not adequate, since PDM systems in general do not have the capability to

distinguish between properties related to parts and properties as a general characterization criteria independent from the actually present product data population.

4   Part Structure and Relationships

The STEP PDM Schema 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.

The PDM Schema 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.

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, the STEP PDM Schema 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.

4.1   Explicit Assembly Bill Of Material

The STEP PDM Schema supports explicit hierarchical assembly structures representing assemblies and the constituents of those assemblies. A constituent of an assembly may itself also be an assembly (a subassembly), which in turn is made up of constituents. Detail components make up the leaves of these hierarchical part structures. This explicit part structure corresponds to the traditional engineering and manufacturing bill of material parts list.

Each individual node in this hierarchy is represented by a part master identification. The relationships are made between product_definition entities representing a view definition of the part master. The relationship itself represents the usage occurrence of a constituent definition within the immediate parent assembly definition. In the simplest form, this structure corresponds to a basic indentured parts list (see Figure 2).

Part List

- Hub Assembly
  - disc with holes
  - cap
  - sleeve sub-assembly
    - gasket
    - cylinder
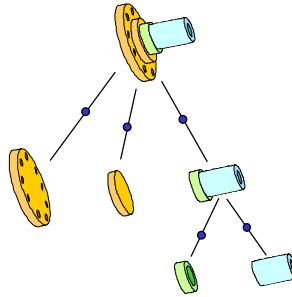
Figure 2 : Basic Indentured Part List

Each node in the assembly structure breakdown is a part master identification, representing the definitional template for the part. The nodes in the product structure each have a corresponding entry in the part list, with indenture level corresponding to depth in the structure.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of basic explicit assembly structure are shown in Diagram 19.

defined view of
assembly version

part identification
.product_definition #3

relating_product_
definition

defined view of
Sub-assembly version

product_definition_relationship =>
product_definition_usage =>
assembly_component_usage =>
next_assembly_usage_occurrence

id = sub assembly occurrence id
name = 'single instance usage'
description = display name
reference_designator = $

part identification
.product_definition #2

related_product_
definition

relating_product_
definition

defined view of
component version

product_definition_relationship =>
product_definition_usage =>
assembly_component_usage =>
next_assembly_usage_occurrence

id = component occurrence id
name = 'single instance usage'
description = display name
reference_designator = $

part identification
.product_definition #1

related_product_
definition

Diagram 19 : Explicit Assembly Instance Diagram

4.1.1.1 assembly_component_usage

This entity represents the general relationship between two part master identifications, one a definition of a component and the other a definition of the parent assembly. This entity is normally not recommended to be instantiated as itself, but is typically instantiated as the subtype next_assembly_usage_occurrence. This subtype represents a unique individual occurrence of the component definition as used within the parent assembly. The assembly_component_usage is only instantiated as itself if there is an explicit requirement to represent the general relationship between a component and an assembly definition.

Attributes
- The *id* attribute provides a unique identifier for the relationship.
- The *name* attribute provides a general nomenclature for the relationship.
- The *description* attribute allows for additional text that further may describe the relationship.
- The *relating_product_definition* attribute is a reference to the parent assembly.
- The *related_product_definition* attribute is a reference to the constituent of the assembly.
- The *reference_designator* attribute is optional.

ENTITY assembly_component_usage	Attribute Population	Remarks
id	type : identifier = string	Should be unique in combination with a given relating (parent) and related (child) product_definition.
name	type : label = string	

description	type : text = string	OPTIONAL
relating_product_definition	type : entity = product_definition	reference to assembly
related_product_definition	type : entity = product_definition	reference to component
reference_designator	type : identifier = string	OPTIONAL

Preprocessor Recommendations: No cyclic product_definiton_relationship structures may be defined. The value of the id attribute must be unique in combination with the same relating assembly and related component definitions. It is generally recommended that only subtypes of assembly_component_usage be instantiated. It is not recommended to instantiate the optional reference_designator attribute for an instance of assembly_component_usage. If information on the assembly type is to be represented, then the name attribute of the product_definition referenced via relating_product_definition should be used. Examples of such type information are 'functional assembly', 'manufacturing assembly', and 'design assembly'.

Postprocessor Recommendations: When importing an instance of assembly_component_usage as itself, postprocessors should interpret this as representing the general assembly-component relationship.

Related Entities: None specific.

4.1.1.2 next_assembly_usage_occurrence

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. The id attribute contains a unique instance identifier for the individual component occurrence.

Preprocessor Recommendations: No cyclic product_definiton_relationship structures may be defined. The value of the id attribute must be unique between the same assembly and component definitions. The name attribute may be used to record the item find number corresponding to the particular instance of the component as called out on an engineering drawing. It is recommended the description attribute contain an instance display name label for the usage occurrence, if one exists. The reference designator attribute is optional: it may be used to represent a positional designation for the component within the assembly.

For AP214 compatibility, preprocessors may generate an instance of product_definition_context_association to relate a product_definition representing an assembly to a product_definition_context named 'assemblydefinition'. The presence of such an additional context named 'assembly definition' indicates that the product_definition has component structure provided beneath it within the STEP exchange file, i.e. it is referenced as relating in an assembly_component_usage relationship. Note that this information is inherent in the exchange file – in general it may be obtained by simply traversing the product structure provided in the file.

Postprocessor Recommendations: When importing an instance of next_assembly_usage_occurrence as itself, postprocessors should interpret this to represent a single individual usage occurrence of the component definition within the parent assembly definition.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context and life cycle stage */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');

/* optional AP214 specific characterization for assembly definition */
#340 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
```

```
#520 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);

/* type discriminator for part as product */
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #440));

/* part master for component definition */
#360 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#380));
#380 = PRODUCT('g1', 'gasket', $, (#220));
#390 = PRODUCT_DEFINITION_FORMATION('A','',#380);
#400 = PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);

/* part master for assemly definition */
#420 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#440));
#440 = PRODUCT('s1', 'sleeve assembly', $, (#220));
#450 = PRODUCT_DEFINITION_FORMATION('A','',#440);
#460 = PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly',
#450, #230);

/* single usage occurrence of component defn within assembly defn */
#530 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu1', 'single instance usage',
   'gasket usage 1', #460, #400, $);
```

Example 16 : exchange file for explicit assembly

4.1.2 Quantified Component Usage

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 in two ways in the PDM Schema:

- quantified usage occurrence,
- multiple individual usage occurrences.

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

The PDM Schema uses separate single usage occurrences to represent multiple occurrences of a component within an assembly when the various components must be distinguished individually (see 4.1.3).

The quantified usage occurrence associates a quantity value with the multiple use of a component definition in the parent assembly definition. The quantified usage occurrence simply relates a quantity value with the multiple component occurrences; it does not distinguish the individual occurrences independently. The quantified usage occurrence is most commonly used for standard component parts, such as fasteners, that need to be quantified but do not need to be individually distinguished. This structure corresponds to a basic indentured parts list with the associated part quantity values (see Figure 3).

Part List   Quantity
- Hub Assembly
    – disc with holes   1
    – cap   1
    – sleeve sub-assembly   1
        • cylinder   1
        • gasket   2
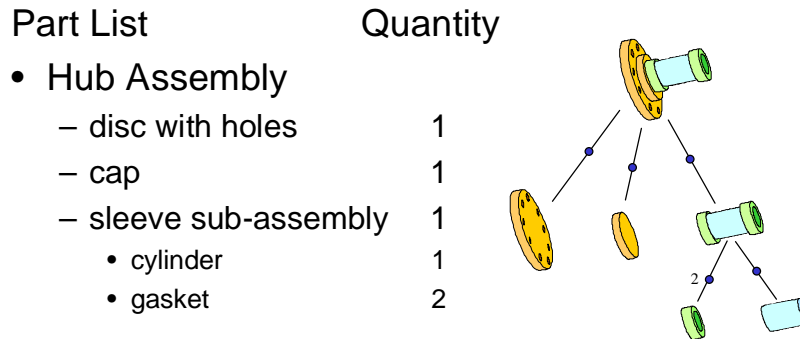
Figure 3 : Indentured Parts List with Associated Quantity

As with the basic indentured parts list, each node in the part structure breakdown is a part master, representing the definitional template for the component or assembly. The nodes of the product structure each have a corresponding entry in the part list, with the indenture level corresponding to depth in the assembly structure. The quantity of each usage occurrence is identified in the parts list - the single occurrence usage has an implicit quantity of one. In this example, the relationship between the gasket component definition and the parent sleeve subassembly is represented by a quantified component usage to denote two occurrences of the gasket definition used in the parent sleeve subassembly.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements for quantified assembly structure are shown in Diagram 20.
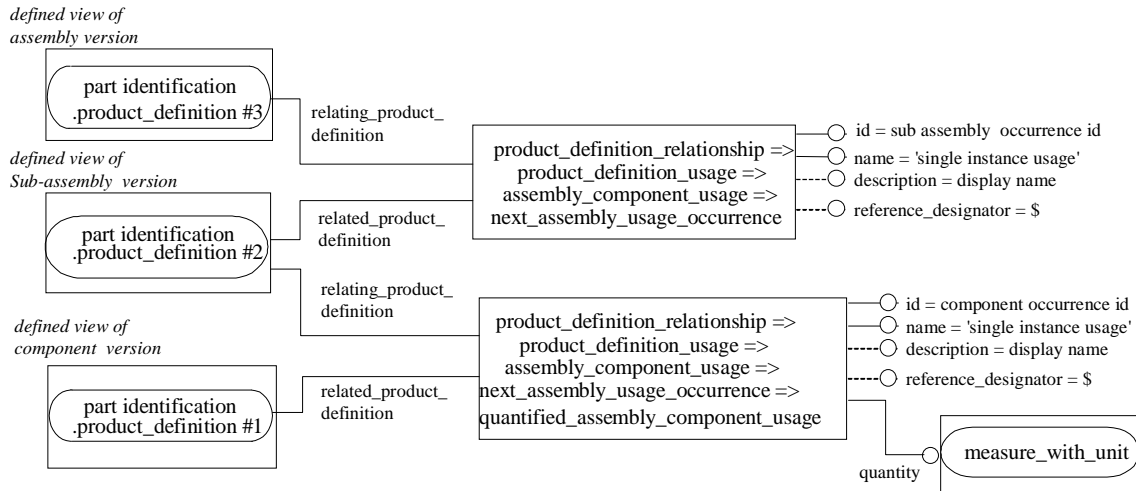
Diagram 20 : Quantified Assembly Instance Diagram

4.1.2.1 quantified_assembly_component_usage

This entity is a subtype of assembly_component_usage. When the quantified_assembly_component_usage is required, it should be created as an AND instantiation with the next_assembly_usage_occurrence. It adds the attribute quantity to identify the number of occurrences of the constituent definition that are used in the parent assembly. With the quantified_assembly_component_usage, multiple individual occurrences of the constituent part are not distinguished independently.

Attributes
- The *quantity* attribute gives the number of usage occurrences of a component within an assembly.

ENTITY quantified_assembly_component _usage	Attribute Population	Remarks
id	see supertype	
name	see supertype	
description	see supertype	OPTIONAL
relating_product_definition	see supertype	reference to assembly
related_product_definition	see supertype	reference to component
reference_designator	see supertype	OPTIONAL should not be instaniated
quantity	type : entity = measure_with_unit	

Preprocessor Recommendations: This entity should be instantiated when quantities greater than one need to be represented, but when the individual occurrences do not need to be independently distinguished. This is created as a "complex" instance of next_assembly_usage_occurrence AND quantified_assembly_component_usage. For a quantity of one, a simple instance of next_assembly_usage_occurrence alone may be used. However, quantified_assembly_component_usage needs to be used if the number of occurrences is specified by a specific unit, e.g. the usage of "one liter of oil" as a component in an assembly.

Postprocessor Recommendations: When importing an instance of next_assembly_usage_occurrence AND quantified_assembly_component_usage, a quantity value of one should be interpreted as equivalent to the simple next_assembly_usage_occurrence.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context and life cycle stage */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');


/* type discriminator for part as product */
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #440));

/* part master for component definition */
#360 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#380));
#380 = PRODUCT('g1', 'gasket', $, (#220));
#390 = PRODUCT_DEFINITION_FORMATION('A','',#380);
#400 = PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);

/* part master for assemly definition */
#420 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $,(#440));
#440 = PRODUCT('s1', 'sleeve assembly', $, (#220));
#450 = PRODUCT_DEFINITION_FORMATION('A','',#440);
#460 = PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly',
#450, #230);

/* quantified multiple use occurrence of component within assembly */
#860 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
#870 = NAMED_UNIT(#860);
#880 = MEASURE_WITH_UNIT(COUNT_MEASURE(2), #870);
#890 = (ASSEMBLY_COMPONENT_USAGE($) NEXT_ASSEMBLY_USAGE_OCCURRENCE()
   PRODUCT_DEFINITION_RELATIONSHIP('guQ', 'quantified instance usage',
   'quantified usage of gasket', #460, #400) PRODUCT_DEFINITION_USAGE()
   QUANTIFIED_ASSEMBLY_COMPONENT_USAGE(#880));
```

Example 17: exchange file for quantified component usage

Illustrated Example: Parts Lists and Item Find Number

The following example illustrates the mechanical drawing and corresponding parts list for two typical example assembly parts, PCB A and PCB B. Each drawing illustrates the use of the item find number.
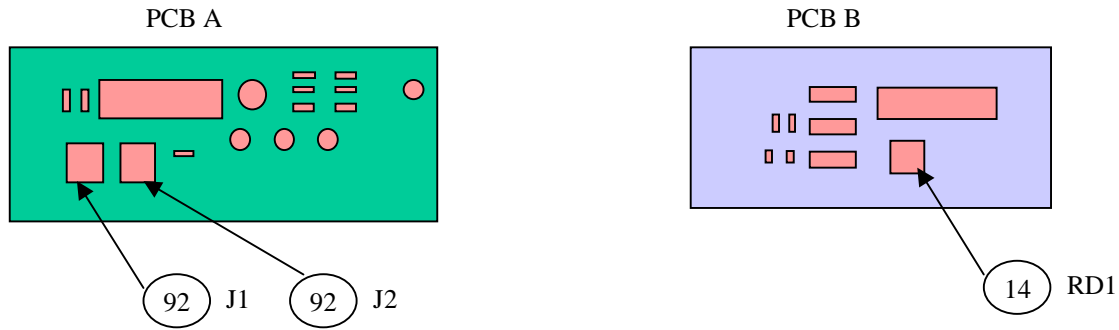
Parts list for PCB A

Find number	Part number	Description	Remarks	Quantity
90	con1001	Connector		1
91	scr2101	Screw		3
92	mem801	Memory	J1	1
92	mem801	Memory	J2	1
…….				

Parts list for PCB B

Find number	Part number	Description	Remarks	Quantity
13	res225	Resistor		1
14	mem801	Memory	RD1	1
…….				

Figure 4 : Illustrated example of drawings, associated parts lists, and item find numbers

In the example above, the same component (part number mem801) is used in both assemblies PCB A and PCB B. However, the item find number is different for each assembly. This number is used to identify components in drawings it assigns to the component a name that is specific to the assembly in which it is being used. In the PDM Schema, the item find number may be assigned to the name attribute of next_assembly_usage_occurrence. The same attribute should be used if the part occurrence is instantiated as quantified_assembly_component_usage.

4.1.3 Multiple Individual Component Occurrences

The basic explicit assembly represents a single occurrence of the component definition within the assembly definition. Multiple occurrences of a constituent used within an assembly are represented in one of two ways in the PDM Schema:

- quantified usage occurrence,
- multiple individual usage occurrences.

The quantified usage occurrence identifies the component quantity, but does not allow independent identification of the individual occurrences when a component definition is used multiple times (see 4.1.2).

The PDM Schema 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.

For example, consider a component that is a flexible pipe. The geometrical representation assigned to the definition of that component may be a simple I shaped tube. The geometrical representation associated with one occurrence of the pipe positioned in a parent assembly may be an S shape, as constrained by the assembly environment. A second occurrence of the pipe definition may be used in the same parent assembly, but in a different position. The geometrical representation assigned to this occurrence may be an L shape due to different constraints from the different position of the component within the parent assembly.

This structure corresponds to a basic indentured parts list with the associated part quantity values where the individual component occurrences are individually identified (see Figure 5).
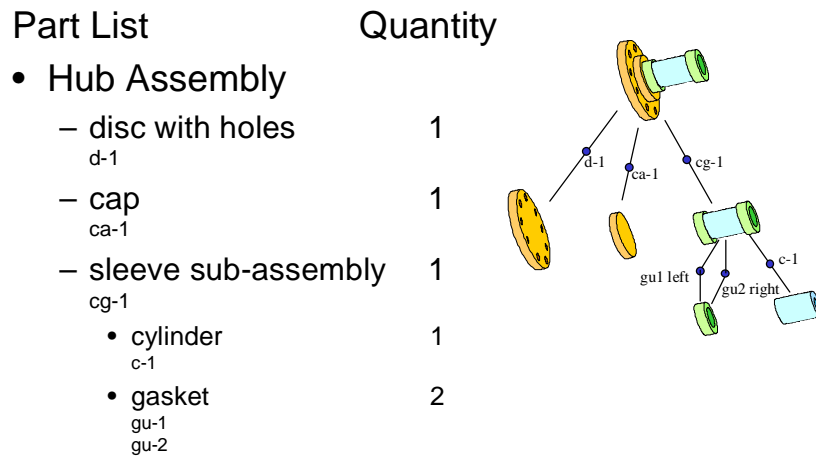
Figure 5 : Indentured Parts List with Multiple Individual Components

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of individually distinguished occurrences of a constituent in an assembly structure are illustrated in Diagram 21.
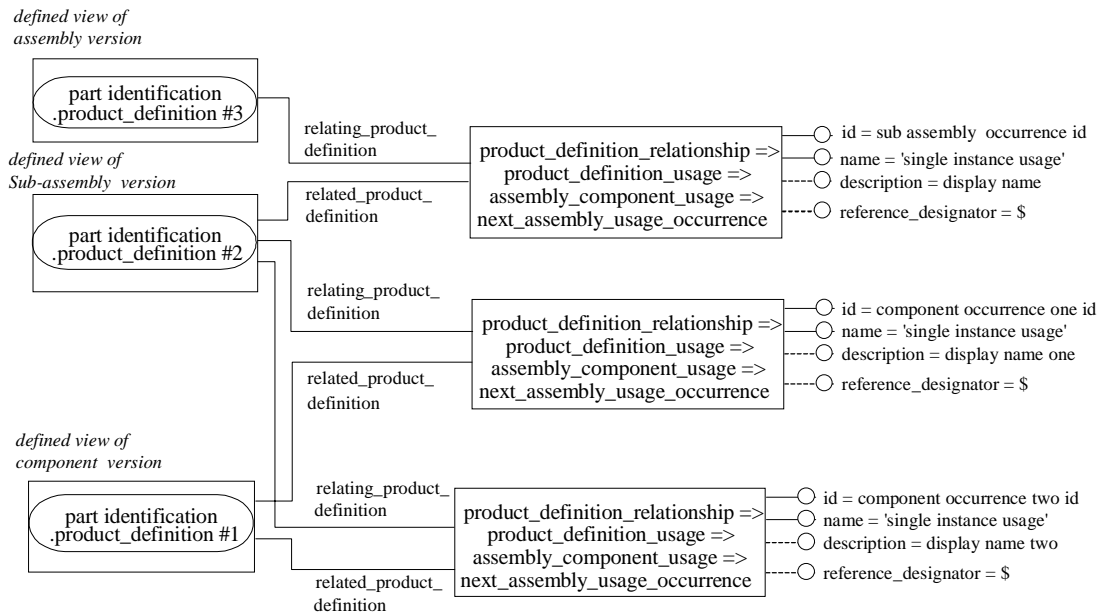
Diagram 21 : Multiple Individual Component Usage Instance Diagram

Preprocessor Recommendations: The entity next_assembly_usage_occurrence (see 4.1.1.2) should be instantiated once for each individual occurrence of a component definition used in the parent assembly. The value of the attribute id must be unique for each instance. The description attribute may contain a unique display name for presentation of the assembly structure.

Postprocessor Recommendations: The sum of single instances of next_assembly_usage_occurrence that exist between an assembly definition and a component definition represents the total quantity of component occurrences used in the parent assembly.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context and life cycle stage */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');

/* optional AP214 specific characterization for assembly definition */
#340 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
#520 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
```

```
/* type discriminator for part as product */
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #440));

/* part master for component definition */
#360 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#380));
#380 = PRODUCT('g1', 'gasket', $, (#220));
#390 = PRODUCT_DEFINITION_FORMATION('A','',#380);
#400 = PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);

/* part master for assemly definition */
#420 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#440));
#440 = PRODUCT('s1', 'sleeve assembly', $, (#220));
#450 = PRODUCT_DEFINITION_FORMATION('A','',#440);
#460 = PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly',
#450, #230);

/* multiple single occurrences of component defn in assembly defn */
#530 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu-1', 'single instance usage',
    'gu1 left', #460, #400, $);
#540 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu-2', 'single instance usage',
    'gu2 right', #460, #400, $);
```

Example 18: exchange file for multiple individual component usages

4.1.4 Promissory Component Usage

The STEP PDM Schema supports promissory usage relationships between components and some higher-level assembly that is not the immediate parent in the hierarchy. This may be important to relate an outsourced component to a top-level assembly definition related to a configuration and product_concept. A supplier may only require the configuration and end item identification that is associated with the component of interest, and may not need to know the detailed assembly structure in between.   The promissory component usage may be useful during early design phases to create preliminary BOMs for prototyping.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of promissory assembly are illustrated in Diagram 22.
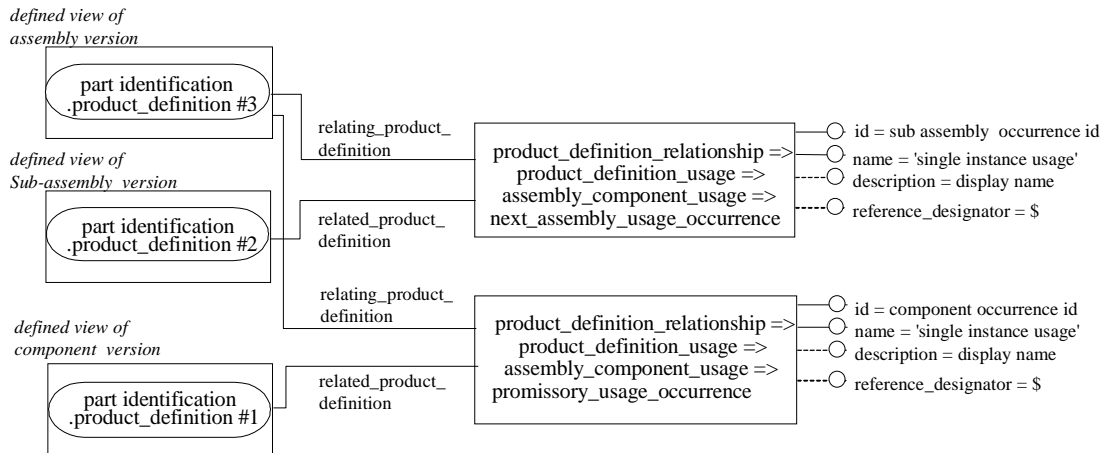
Diagram 22: Promissory Assembly Usage Instance Diagram

4.1.4.1 promissory_usage_occurrence

This entity is a subtype of assembly_component_usage. It represents the usage occurrence of a component within a higher-level assembly that is not the immediate parent, in the case where the detailed assembly structure in between the component and the higher-level assembly is not represented.   A promissory_usage_occurrence specifies the intention to use the constituent in an assembly. It may also be used when the product structure is not completely defined.

Preprocessor Recommendations: The higher-level assembly should be associated with a configuration item with its related end item identification in the typical usage case for the promissory_usage_occurrence. If the assembly structure between the component and the higher-level assembly must be specified, the next_assembly_usage_occurrence should be used (see 4.1.1.2).

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context and life cycle stage */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10,'');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');
```

```
/* optional AP214 specific characterization for assembly definition */
#340 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
#520 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);

/* type discriminator for part as product */
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #440));

/* part master for component definition */
#360 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#380));
#380 = PRODUCT('g1', 'gasket', $, (#220));
#390 = PRODUCT_DEFINITION_FORMATION('A','',#380);
#400 = PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);

/* part master for top level assemly definition */
#420 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#440));
#440 = PRODUCT('h1', 'hub assembly', $, (#220));
#450 = PRODUCT_DEFINITION_FORMATION('A','',#440);
#460 = PRODUCT_DEFINITION('hv1', 'design view on hub assembly', #450,
#230);

/* promissory usage occurrence of component in top level assembly */
#530 = PROMISSORY_USAGE_OCCURRENCE('pu-1', 'promissory instance usage',
'pu-1 promissory', #460, #400, $);
```

Example 19: exchange file for promissory assembly usage

4.2 *Multi-Level Assembly Digital Mock Up*

STEP has the capability to identify individual occurrences of component in a multi-level assembly (see 4.1.3). This provides the ability to assign to each occurrence an identifier, a position in the assembly, a geometrical representation, or other properties that may be different from that assigned to the part definition of the component.

In order to distinguish a specific occurrence of a component in an assembly of more than two hierarchical levels, the specified_higher_usage_occurrence entity is used. For example, a wheel-axle subassembly is composed of an 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 back. The requirement to individually identify the left-front wheel, for example, is supported by this capability.

From the previous wheel-axle-chassis example, an instance of specified_higher_usage_occurrence is used to represent the left-front wheel. The related_product_definition attribute references the product_definition that represents the part definition of the component wheel. The relating_product_definition attribute references the product_definition representing the definition of the higher-level chassis assembly. The attribute next_usage is a reference to the next_assembly_usage_occurrence that represents the left wheel occurrence in the wheel-axle subassembly.   The attribute upper_usage is a reference to the next_assembly_usage_occurrence representing the front occurrence of the wheel-axle subassembly within the higher-level chassis assembly.

This structure corresponds to an indentured parts list with associated part quantity values and individual identification of the specific component occurrences used within a multi-level assembly (see Figure 6).

Part List   Quantity

- Hub Assembly
  - disc with holes   1
    d-1
  - cap   1
    ca-1
  - sleeve sub-assembly   2
    su-1 *(rear)*
    - cylinder   1
      cu-1 *(rear)*
    - gasket   2
      gu-1a *(rear left)*
      gu-2a *(rear right)*
    su-2 *(front)*
    - cylinder   1
      cu-2 *(front)*
    - gasket   2
      gu-1b *(front left)*
      gu-2b *(front right)*
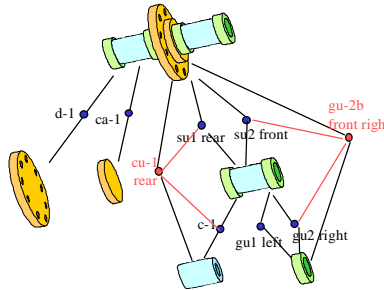
Figure 6 : Indentured Parts List with Specified Higher Component Usage

In Figure 6, the specific higher usage cu-1 of the cylinder (c-1 in a sleeve subassembly) within the rear sleeve (su-1 in a hub assembly) of the higher-level hub assembly is illustrated. Also illustrated is the right gasket (gu2 within a sleeve subassembly) within the front sleeve (su-2 within a hub assembly) of the higher-level hub assembly, individually identified as gu-2b. Each specified higher usage occurrence of the gasket, as well as that of the cylinder, would be individually identified using the same mechanism as illustrated for the rear cylinder and the front right gasket.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of explicit multi-level assemblies and digital mockup are illustrated in Diagram 23.
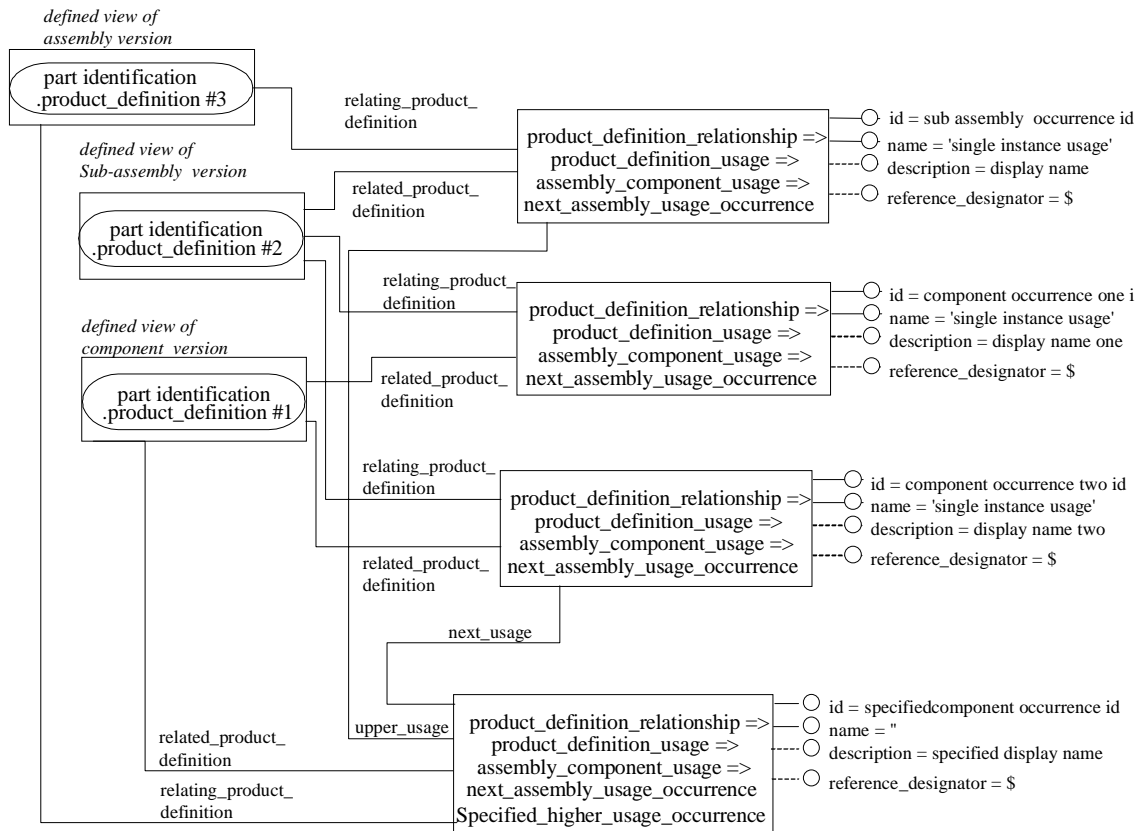
Diagram 23 : Multi-level Assembly DMU Instance Diagram

4.2.1.1 specified_higher_usage_occurrence

This entity is a subtype of assembly_component_usage. This entity represents the specific use occurrence within a higher-level assembly of an individual occurrence of a component definition used in an immediate parent sub-assembly.

EXAMPLE - a wheel-axle subassembly is composed of an 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 back. One specific higher usage of the left wheel occurrence within the higher-level chassis assembly is in the front wheel-axle subassembly   (other is in the rear).   The specified_higher_usage_occurrence entity represents the individually identified left-front wheel. A second instance of this entity could represent another specified higher usage occurrence of the wheel, for example left rear or right front.

In addition to inherited attributes, this entity defines the attributes next_usage and upper_usage to relate a single usage occurrence at one level in the assembly hierarchy (the next_usage) to a single usage occurrence at the next higher level in the assembly (the upper usage).

In the case of a specified usage occurrence at more than one higher level in the assembly hierarchy, the upper_usage attribute will reference the specified_higher_usage instance representing the specified occurrence at the higher level. In this way, instances of specified_higher_usage_occurrence are "chained" together to identify specific occurrences of components in an assembly hierarchy with an arbitrary number of levels.

Attributes
- The *next_usage* attribute identifies the single usage occurrence at a given level in the hierarchy.
- The *upper_usage* attribute identifies the specific higher usage occurrence one level up in the hierarchy.

ENTITY specified_higher_usage_occurrence	Attribute Population	Remarks
id	see supertype	
name	see supertype	
description	see supertype	OPTIONAL
relating_product_definition	see supertype	
related_product_definition	see supertype	
reference_designator	see supertype	OPTIONAL
next_usage	type : entity = next_assembly_usage_occurrence	
upper_usage	type : entity = assembly_component_usage (next_assembly_usage_occurrence OR specified_higher_usage_occurrence)	References s_h_u_o in the case of a specified usage at more than one higher level in the hierarchy

Preprocessor Recommendations: This entity should be instantiated once for each specific use in a higher-level assembly of an individual occurrence of a component definition used in a parent assembly. The value of the attribute id must be unique for each instance. The description attribute may contain a unique display name for presentation of the assembly structure.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context and life cycle stage */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');

/* optional AP214 specific characterization for assembly definitions */
#340 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
#520 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
#530 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#660, #350, #340);

/* type discriminator for part as product */
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #680, #440,
#640));

/* part master for component definitions */
#360 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#380, #680));
#380 = PRODUCT('g1', 'gasket', $, (#220));
#390 = PRODUCT_DEFINITION_FORMATION('A','',#380);
#400 = PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);

#680 = PRODUCT('c1', 'cylinder', $, (#220));
#690 = PRODUCT_DEFINITION_FORMATION('A','',#680);
```

```
#700 = PRODUCT_DEFINITION('cv1', 'design view on cylinder', #690,
#230);

/* part master for sub-assemly definition */
#420 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#440, #640));
#440 = PRODUCT('s1', 'sleeve assembly', $, (#220));
#450 = PRODUCT_DEFINITION_FORMATION('A','',#440);
#460 = PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly',
#450, #230);

/* part master for higher level hub assemly definition */
#640 = PRODUCT('h1', 'hub assembly', $, (#220));
#650 = PRODUCT_DEFINITION_FORMATION('A','',#640);
#660 = PRODUCT_DEFINITION('hv1', 'design view on hub assembly', #650,
#230);

/* single occurrence of component (cylinder) in subassembly defn */
#520 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('cu-1', 'single instance usage',
    'cu1', #460, #700, $);

/* multiple single occurrences of component (gasket) in subassembly */
/* the left gasket in a sleeve subassembly */
#530 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu-1', 'single instance usage',
    'gu1 left', #460, #400, $);
/* the right gasket in a sleeve subassembly */
#540 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu-2', 'single instance usage',
    'gu2 right', #460, #400, $);

/* multiple single occurrences of subassembly defn in assembly defn */
/* the rear sleeve subassembly in a hub assembly */
#550 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('su-1', 'single instance usage',
    'su1 rear', #660, #460, $);
/* the front sleeve subassembly in a hub assembly */
#560 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('su-2', 'single instance usage',
    'su2 front', #660, #460, $);

/* specified usage of individual components in higher level assembly */
/* the right gasket in the front sleeve subassembly */
#570 = SPECIFIED_HIGHER_USAGE_OCCURRENCE('gu-2b', 'single instance
usage', 'gu2b front right', #660, #400, $, #560, #540);
/* the cylinder in the rear sleeve subassembly */
#580 = SPECIFIED_HIGHER_USAGE_OCCURRENCE('cu-1', 'single instance
usage', 'cu1 rear', #660, #700, $, #550, #520);
```

Example 20 : exchange file for multi-level assembly DMU

4.3   Different Views on Assembly Structure

Different view definitions of the same version of a part may participate in different explicit product structures in the PDM Schema. 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 different from the design structure. Finally, a support/as-maintained view of the part version representing the physical part definition that may participate in yet another different disassembly structure. Figure 7 illustrates two different views on a part structure.

Assembly   **Disassembly**
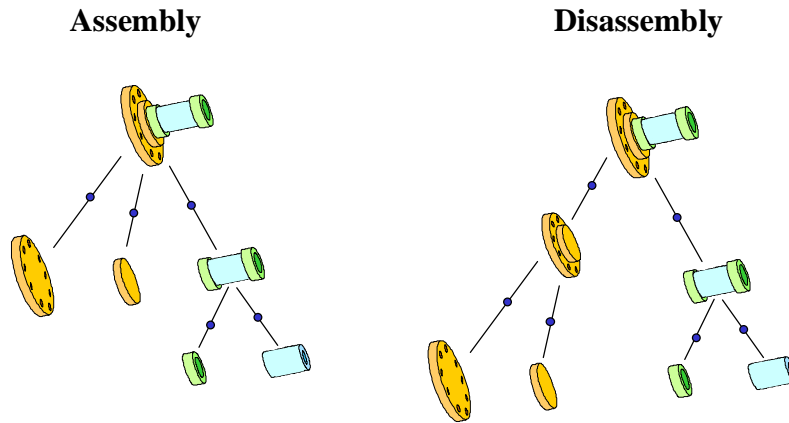
Figure 7 : Different Views on Assembly Structure

Many of the parts that make up the 'nodes' in the two different assembly structures in Figure 7 have the same base identification - they simply have multiple definitions from different life cycle viewpoints. The disc with holes, the cap, and the sleeve sub-assembly are all present in both views of the part structure. However, in this example there is also a new 'node' in the disassembly structure. It corresponds to a sub-assembly, composed of a disc with holes and a cap, which was not present in the assembly structure (see Figure 8).

Assembly List	Qty	Disassembly List	Qty
• Hub Assembly		• Hub Assembly	
– disc with holes	1	– discap sub-assy	1
– cap	1	• disc with holes	1
– sleeve sub-assy	1	• cap	1
• cylinder	1	– sleeve sub-assy	1
• gasket	2	• cylinder	1
		• gasket	2

Figure 8 : Different Indentured Lists Corresponding to Different Structure Views

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and relationships that support the requirements of alternate life cycle viewpoints on product structure are the same as those already described for part structures. Recall that the structural element of the part master 'node' is the product_definition entity, representing a view definition of a part version. The previous description presumed that each part master element in the assembly structure had a consistent life cycle view definition - a single assembly structure from a single view definition context.

In this example of multiple part structures, the part master information corresponding to the 'nodes' disc with holes and cap (and sleeve sub-assy plus it's components) each has two part view definitions. One view definition may represent, for example, the design or manufactured assembly structure; the other a maintenance or disposal disassembly structure. Both of the view definitions are related to the same part version. In this way, the information that is common across the different part structures (the base version identification for all 'nodes' that have definition in both structures) is represented only once. Information that is different between the different views, such as associated management data, property information, or in this case product structure itself is managed independently by using different product_definitions.

The assembly structure would be constructed between the product_definition instances that represent the part view definitions within the same appropriate view context, say for example design.

The disassembly structure would also be constructed between product_definition instances representing part view definitions within one appropriate lifecycle stage - but a different one, for example maintenance. Present only in the disassembly structure, the new or "phantom" part is a sub-assembly composed of a disc with holes and a cap - it is not present in the assembly structure. The part master information corresponding to this discap sub-assy need only have a single part view definition in the appropriate lifecycle stage, e.g. maintenance. The complete disassembly structure is constructed using this product_definition, along with those of the other component part masters for that same view context.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context for design assembly life cycle */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');

/* primary application context for maintenance disassembly view */
#110 = APPLICATION_CONTEXT('maintenance planning');
#240 = PRODUCT_DEFINITION_CONTEXT('part definition', #110,
'maintenance');

/* optional AP214 specific characterization for assembly definitions */
#340 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
#820 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
#830 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#660, #350, #340);

/* type discriminator for part as product */
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #680, #440,
#640));
#360 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#380, #680));
#420 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#440, #640));

/* part master for common components, each with two view definitions */
/* part master for disc with holes - two view definitions */
#380 = PRODUCT('d1', 'disc with holes', $, (#220));
```

```
#390 = PRODUCT_DEFINITION_FORMATION('A','',#380);
#400 = PRODUCT_DEFINITION('dvd1', 'design view on disc', #390, #230);
#410 = PRODUCT_DEFINITION('mvd1', 'maintenance view on disc', #390,
#240);

/* part master for cap component - two view definitions */
#680 = PRODUCT('cap1', 'cap for hub', $, (#220));
#690 = PRODUCT_DEFINITION_FORMATION('A','',#680);
#700 = PRODUCT_DEFINITION('dvc1', 'design view on cap', #690, #230);
#710 = PRODUCT_DEFINITION('mvc1', 'maintenance view on cap', #690,
#240);

/* part master for "phantom" sub-assemly definition - one view only */
#440 = PRODUCT('p1', 'hubcap subassembly', $, (#220));
#450 = PRODUCT_DEFINITION_FORMATION('A','',#440);
#460 = PRODUCT_DEFINITION('mvp1', 'maintenance view hubcap
disassembly', #450, #240);

/* part master for higher level hub assembly - two view definitions */
#640 = PRODUCT('h1', 'hub assembly', $, (#220));
#650 = PRODUCT_DEFINITION_FORMATION('A','',#640);
#660 = PRODUCT_DEFINITION('dvh1', 'design view on hub assembly', #650,
#230);
#670 = PRODUCT_DEFINITION('mvh1', 'maintenance view on hub assembly',
#650, #240);

/* design life cycle 'as planned' assembly structure */
/* single occurrence of component (disc) defn in hub assembly defn */
#520 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('d-1', 'single instance usage',
    'du-1', #660, #400, $);
/* single occurrence of component (cap) defn in hub assembly defn */
#530 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('c-1', 'single instance usage',
    'cu-1', #660, #700, $);

/* maintenance life cycle planned disassembly structure */
/* single "phantom" hubcap subassembly occurrence in hub disassembly */
#560 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('hc-1', 'single instance usage',
    'hc-1', #670, #460, $);
/* single occurrence of component (disc) defn in hubcap sub-assembly */
#540 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('d-1', 'single instance usage',
    'du-1', #460, #410, $);
/* single occurrence of component (cap) defn in hubcap sub-assembly */
#550 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('c-1', 'single instance usage',
    'cu-1', #460, #710, $);
```

Example 21: exchange file for different assembly and disassembly structures

4.4   Relating Part Shape Properties to Product Structure

The PDM Schema allows to link geometric structures that result from relating different shape_representations with associated product structure when applicable, i.e. when the geometric structure directly corresponds to the assembly structure.

Two alternatives for the implementation of geometric structures related to assembly structures are recommended:

1. The assembly is described with the components built in. With this approach the shape of the component is mapped into the shape of the assembly via mapped_item. The basic idea of the mapped_item is: an item will become part of another item. The assembly component geometry is used as a template in the assembly geometry.
2. The components of an assembly are described together with the construction history. This approach uses the above-described representation_relationship_with_transformation. The transformation describes the relation between different workspaces.

The usage of both alternatives is considered reasonable, because both mechanisms make sense even in mixed combinations. With regard to the transformations in the context of assembly, a part is in principle incorporated in the assembly only by rigid motion (i.e. translation and/or rotation) excluding mirroring and scaling.

4.4.1 Explicit Representation of Complete Assembly Geometry

This approach represents the assembled model completely. It is appropriate for explicit representation of assemblies. It uses *mapped_item* in combination with *repesentation_map*, which takes a representation and turns it into an item in a second representation. The transformation to be applied is determined from the mapping origin and the mapping target, which are items in the two representations (and therefore founded in the two contexts).

Mapped_item is constrained such that:
- No mapped item shall be dependent on itself to define the representation being mapped (acyclic).
- The mapping origin shall be in the context of the mapping_source representation.

There is no constraint to prevent both representations having the same context (i.e. the new item is shifted in position but does not change spaces). This allows the use of *mapped_item* to position sub-models in a single coordinate space. It is even possible to define an identity matrix transformation by referring to the same entity instance as both mapping_origin and mapping_target. This represents the case where a component is defined in (one of) its final position(s). Until it is mapped into the second representation, it is not included in the representation even though defined in the same space.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of explicit representation of assembly structure with associated external geometry model structure are illustrated in Diagram 24.
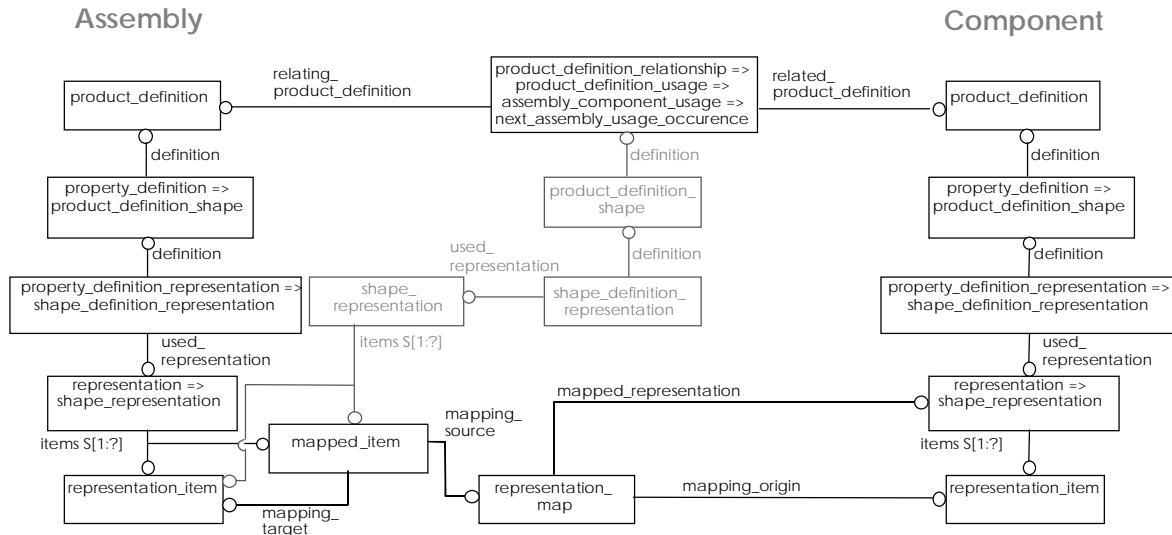
Diagram 24: Assembly Structure with Component Templates Instance Diagram

4.4.1.1 mapped_item

The mapped_item allows the use of a representation as a representation_item, e.g. it allows for the definition of representation using other representations. Thus a shape_representation for an assembly component can participate as a template instance in the definition of the shape of the overall assembly. The representation is defined by mapped_item.mapping_source.mapped_representation. The mapping is achieved through an operator that is a mapping of the mapped_item. mapping_source.mapping_origin onto the mapped_item.mapping_target

Attributes
- The *mapping_source* attribute points to an instance of representation_map that specifies the to be mapped representation.
- The *mapping_target* attribute is an representation_item that defines.

ENTITY mapped_item	Attribute Population	Remarks
mapping_source	type: entity = representation_map	
mapping_target	type: entity = representation_item	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

4.4.1.2 representation_map

Attributes
- The *mapping_origin* attribute.
- The *mapped_representation* attribute references the to be mapped representation.

ENTITY representation_map	Attribute Population	Remarks
mapping_origin	type: entity = representation_item	should be an axis_placement in the given context
mapped_representation	type: entity = representation	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

4.4.2 Implicit Relationships Between Assembly Components

This approach represents the components and the set of instructions how to build the assembly. This approach is adequate for the implicit representation of assemblies.   Thus, representation_ relationship_with_transformation is used to define the relative positions within an assembly. It does not allow for:

- Inclusion - the decision to create a representation, which brings all the components together, is left to the receiving system.
- The ability to define a component in one position and then replicate it in the same space.

The relationships between the assembly and the component on the product_definition level and the shape_representation level has to be linked through a context_dependent_shape_representation. This is necessary to distinguish between several occurrences of the same component within an assembly. According to Part 43 the attributes rep_1 and rep_2 of representation_relationship_ with_transformation are determined as:

rep 1   is defined as the representation with a context to which the transformation applies (in other words rep_1 is defined as the representation with a context to which the transformation can be applied in case of building an assembly representation) and

rep 2   is the representation with a context which is the result of the transformation (in other words rep_2 is defined as the representation with the context into which the component representation has to be transformed in case of building an assembly representation).

Based on this definition the attributes of representation_ relationship_with_transformation should be instantiated as follows:

- representation_relationship_with_transformation.rep_1 for the relation to the shape_ representation for the product_definition that is related by assembly_component_usage as related_product_ definition (identifying the component) and
- representation_relationship_with_transformation.rep_2 for the relation to the shape_ representation for the product_definition that is related by assembly_component_usage as relating_ product_definition (identifying the assembly).

Further restrictions are implied by the Part 43 informal proposition on the entity representation_relationship_with_transformation: When   the   transformation   is   an item_defined_transformation, the ordering of the representations given for the inherited attributes of representation_relationship shall be consistent with the ordering of the two representation_items given as attributes of item_defined_transformation.   Therefore the following usage of the attributes transform_item_1 and transform_item_2 of item_defined_transformation is recommended:

- item_defined_transformation.transform_item_1 for the relation to the representation_ item (e.g. axis2_placement_3d) for the product_definition that is related by assembly_component_usage as related_product_definition (identifying the component) and

- item_defined_transformation.transform_item_2 for the relation to the representation_item (e.g. axis2_placement_3d) for the product_definition that is related by assembly_component_usage as relating_product_definition (identifying the assembly).

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of relating assemblies and components using item_defined_transformation are illustrated in Diagram 25.
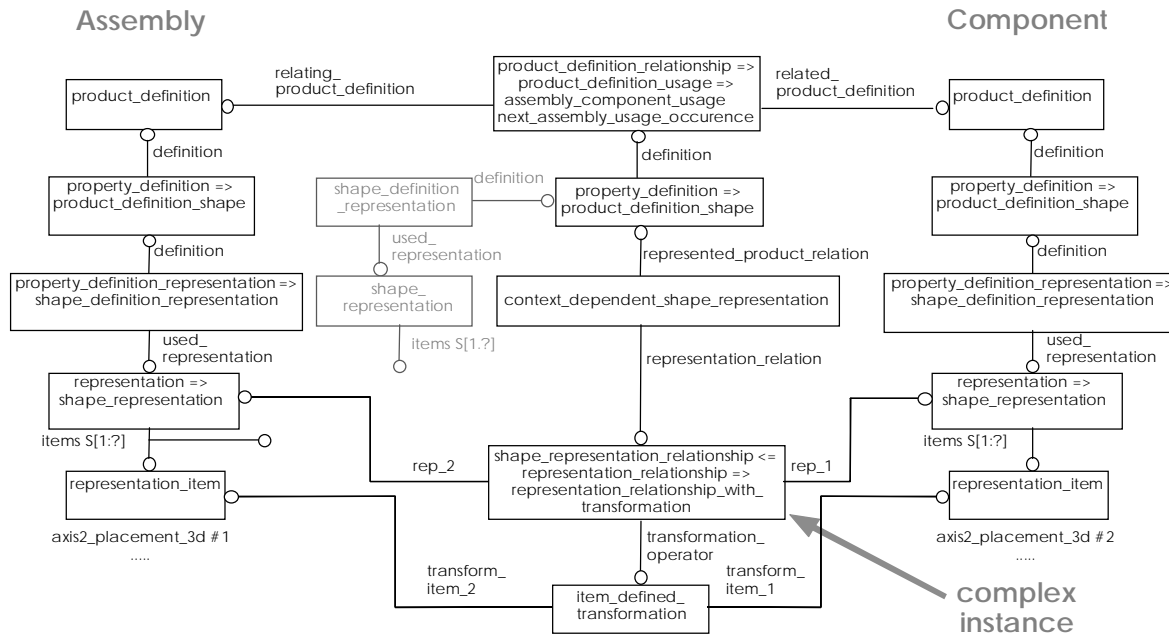
**Diagram 25: Assembly Structure with Defined Component Relationships
(usage of item_defined_transformation)**

Another possibility is the definition of the assembly construction history via a cartesian_ transformation_operator. The Part 42 definition of cartesian_transformation_operator also allows for scaling and mirroring. But scaling and mirroring shall not to be applied in the context of assemblies. Therefore

- the scaling value specified by the attribute cartesian_transformation_operator.scale shall be empty (due to being optional; default scaling is equal 1.0) or shall be 1.0 and

- the determinant of the transformation matrix u (computed from the attributes axis_1, axis_2 and axis_3 in case of using cartesian_transformation_operator_3d by the function base_axis) shall equal 1.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of relating assemblies and components using cartesian_transformation_operator are illustrated in Diagram 26.
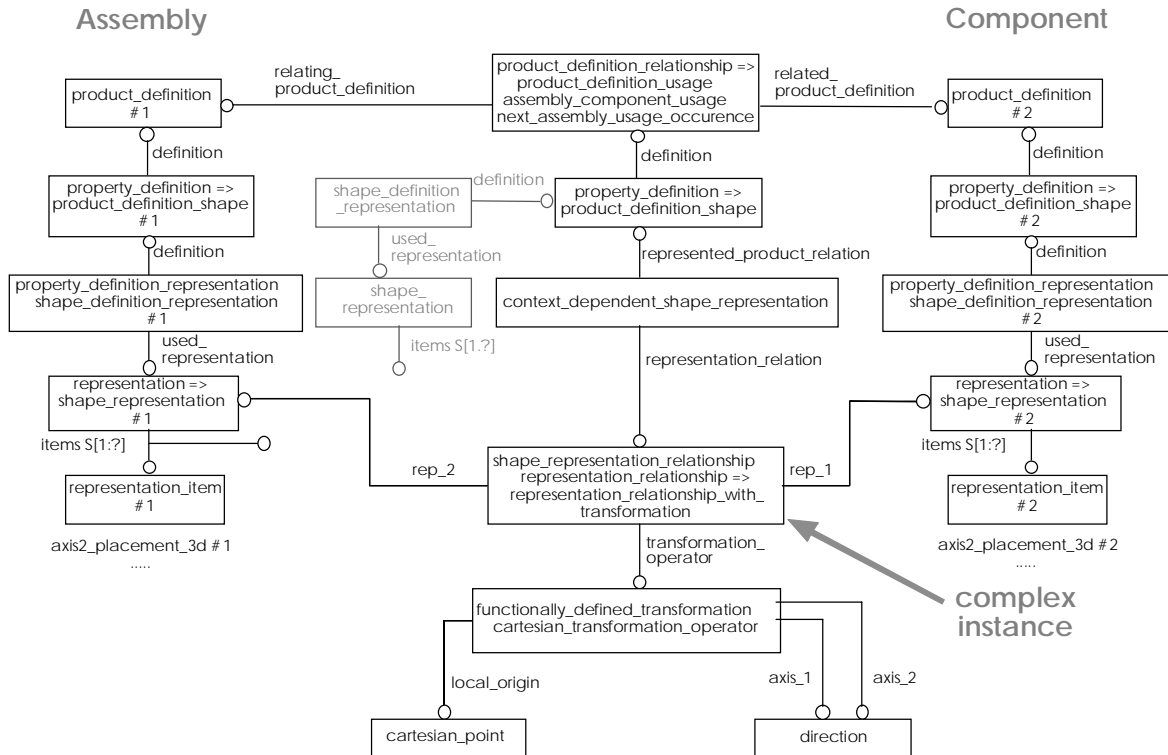
**Diagram 26: Assembly Structure with Defined Component Relationships
(usage of cartesian_transformation_operator)**

4.4.2.1 context_dependent_shape_representation

A context_dependent_shape_representation associates a shape_representation_relationship with a product_definition_shape. In the given context this allows the explicit specification of a shape of the assembly 'as assembled'. Since elements when assembled might change their shape – e.g. under pressure – this representation may differ from the geometric assembly of the individual shapes.

Attributes
- The *representation_relation* attribute is a reference to the shape_representation_relationship used to define the assembly.
- The *represented_product_relation* attribute points to the product_definition_shape that is related to the assembly_usage_occurrence.

ENTITY context_dependent_shape_repres entation	Attribute Population	Remarks
representation_relation	type: entity = shape_representation_relationship	
represented_product_relation	type: entity = product_definition_shape	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

4.4.3 Complete instantiation example for part structure with shape properties

The example file contains two independent products. On part is an assembly of two components ('part 1' and part 2'). For this product structure a corresponding relationship that geometrically relates the components with the assembly is defined. It is assumed that the geometry is defined in external files. For 'part 2' this external file constitutes a managed document that is associated to the focused view of the part. For 'part 1' the external geometry is linked in via an external file for which no revision control exists. Such a flat file association for external geometry would be typical for files directly received from CAD systems.
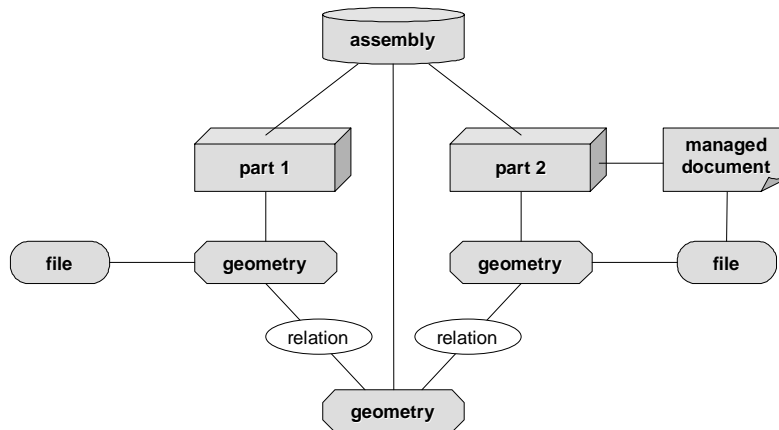
Figure 9 : Schematic Overview of Complete Part Structure with Shape Properties

For the third part in the file two portions of shape are represented and identified independently. For these portions a geometric relationship is defined. This geometric relationship has no correspondence in product structure.
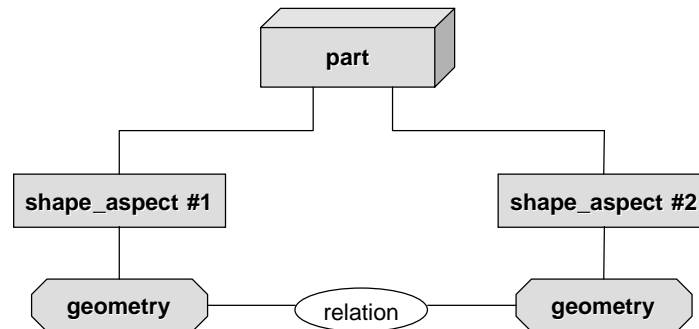
Figure 10 : Schematic Overview of Geometric Relationship

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''), '2;1');
FILE_NAME('', '30.06.1999, 12:05:33', ('N:N:'), (''),
  '', '', '');
```

```
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;

/* Entities #10 to #350 describe meta information and product structure
*/

#10 = APPLICATION_CONTEXT('');
#20 = PRODUCT_CONTEXT('', #10, '');
#30 = PRODUCT('ass', 'assembly', '', (#20));
#40 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#30));
#50 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#30));
#60 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #50, #40);
#70 = PRODUCT('p', 'part', '', (#20));
#80 = PRODUCT('p1', 'part1', '', (#20));
#90 = PRODUCT('p2', 'part2', '', (#20));
#100 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#30, #80, #90,
#70)
    );
#110 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema',
1998
    , #10);
#120 = PRODUCT_DEFINITION_CONTEXT_ROLE('', $);
#130 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, '');
#140 = PRODUCT_DEFINITION_FORMATION('1', '', #30);
#150 = PRODUCT_DEFINITION('assv', 'view on assembly', #140, #130);
#160 = APPLICATION_CONTEXT('mechanical design');
#170 = PRODUCT_DEFINITION_CONTEXT('', #160, 'design');
#180 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#150, #170, #120);
#190 = PRODUCT_RELATED_PRODUCT_CATEGORY('Detail', $, (#80, #90, #70));
#200 = PRODUCT_RELATED_PRODUCT_CATEGORY('Part', '', (#80, #90, #70));
#210 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #200, #190);
#220 = PRODUCT_DEFINITION_FORMATION('1', '', #80);
#230 = PRODUCT_DEFINITION('p1v', 'view on part1', #220, #130);
#240 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#230, $, #120);
#250 = PRODUCT_DEFINITION_FORMATION('1', '', #70);
#260 = PRODUCT_DEFINITION('p', 'view on part', #250, #130);
#270 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#260, $, #120);
#280 = PRODUCT_DEFINITION_FORMATION('1', '', #90);
#290 = PRODUCT_DEFINITION('p2v', 'view on part2', #280, #130);
#300 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#290, $, #120);
#310 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('p1_usage',
    'single instance usage', '', #150, #230, $);
#320 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#330 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
#340 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#150, #330, #320);
#350 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('p2_usage',
    'single instance usage', '', #150, #290, $);

/* Entities #370 to #497 define a manged document which */
/* contains a representation of the shape of 'part2'     */
#370 = PRODUCT('p2_shape_doc', 'document reflecting
        shape of part2', '', (#20));
#380 = PRODUCT_RELATED_PRODUCT_CATEGORY('document', '', (#370));
#390 = DOCUMENT_TYPE('configuration controlled document version');
#400 = DOCUMENT('', '', '', #390);
#410 = APPLIED_DOCUMENT_REFERENCE(#400, 'equivalence', (#290));
```

```
#420 = OBJECT_ROLE('mandatory', '');
#430 = ROLE_ASSOCIATION(#420, #410);
#440 = PRODUCT_DEFINITION_FORMATION('1', '', #370);
#450 = DOCUMENT_PRODUCT_EQUIVALENCE('equivalence', '', #400, #440);
#460 = PRODUCT_DEFINITION_CONTEXT('digital document definition', #10,
''
   );
#470 = PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS('p2_shape_d',
   'digital document for part2 shape', #440, #460,(#475));
#475 = DOCUMENT_FILE('p2_shape_file', '', '', #485, ' ', ' ');
#480 = DOCUMENT_REPRESENTATION_TYPE('digital',#475);
#485 = DOCUMENT_TYPE('');
#490 = IDENTIFICATION_ROLE('document_source',$);
#495 = APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT('',#490,#497,(#475));
#497 = EXTERNAL_SOURCE(IDENTIFIER('part1_geometry.stp'));


/* Entities #500 to #560 define the geometric model for 'part1' */
#500 = PRODUCT_DEFINITION_SHAPE('shape_p1',$,#230);
#510 = SHAPE_DEFINITION_REPRESENTATION(#500,#520);
#520 = SHAPE_REPRESENTATION('sol1',(#570),#530);
#530 = GEOMETRIC_REPRESENTATION_CONTEXT('c1','external',3);
#540 = CARTESIAN_POINT('', (1.0E+001, 0.0E+000, 1.0E+001));
#550 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#560 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#570 = AXIS2_PLACEMENT_3D('', #540, #550, #560);

/* Entities #575 to #585 state that the shape_representation #520 */
/* is externally defined in a digital file with the location */
/* 'part2_geometry.stp'                                       */
#575 = APPLIED_DOCUMENT_REFERENCE(#576,'',(#230));
#576 = DOCUMENT_FILE('p1_shape','',$,#577,'','');
#577 = DOCUMENT_TYPE('geometry');
#578 = ROLE_ASSOCIATION(#579,#575);
#579 = OBJECT_ROLE('mandatory',$);
#580 = DOCUMENT_REPRESENTATION_TYPE('digital',#576);
#581 = IDENTIFICATION_ROLE('document_source',$);
#582 = APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT('',#581,#583,(#576));
#583 = EXTERNAL_SOURCE(IDENTIFIER('part2_geometry.stp'));
#584 = PROPERTY_DEFINITION('external definition',$,#576);
#585 = PROPERTY_DEFINITION_REPRESENTATION(#584,#520);

/* Entities #600 to #660 define the geometric model for 'part2' */
#600 = PRODUCT_DEFINITION_SHAPE('shape_p2',$,#290);
#610 = SHAPE_DEFINITION_REPRESENTATION(#600,#620);
#620 = SHAPE_REPRESENTATION('sol2',(#670),#630);
#630 = GEOMETRIC_REPRESENTATION_CONTEXT('c2','external',3);
#640 = CARTESIAN_POINT('', (3.0E+001, 0.0E+000, 1.0E+001));
#650 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#660 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#670 = AXIS2_PLACEMENT_3D('', #640, #650, #660);

/* Entities #680 and #690 indicate that the shape_representation #620
*/
/* is externally defined by the file represented by #475
*/
#680 = PROPERTY_DEFINITION('external definition',$,#475);
```

```
#690 = PROPERTY_DEFINITION_REPRESENTATION(#680,#620);


/* Entities #700 to #720 defines the shape and two related portions */
/* for the part 'part' #260                                         */
#700 = PRODUCT_DEFINITION_SHAPE('shape_p',$,#260);
#710 = SHAPE_ASPECT('aspect1',$,#700,.T.);
#720 = SHAPE_ASPECT('aspect2',$,#700,.T.);

/* Entities #730 to #790 define the geometry related to  */
/* shape_aspect #710                                     */
#730 = PROPERTY_DEFINITION('shape for aspect1',$,#710);
#740 = SHAPE_DEFINITION_REPRESENTATION(#730,#750);
#750 = SHAPE_REPRESENTATION('sa1',(#795),#760);
#760 = GEOMETRIC_REPRESENTATION_CONTEXT('ca1','external',3);
#770 = CARTESIAN_POINT('', (3.0E+001, 0.0E+000, 1.0E+001));
#780 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#790 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#795 = AXIS2_PLACEMENT_3D('', #770, #780, #790);
/* the link to the external definition in a file has been omitted here
*/

/* Entities #800 to #860 define the geometry related to  */
/* shape_aspect #720                                     */
#800 = PROPERTY_DEFINITION('shape for aspect2',$,#720);
#810 = SHAPE_DEFINITION_REPRESENTATION(#800,#820);
#820 = SHAPE_REPRESENTATION('sa2',(#870),#830);
#830 = GEOMETRIC_REPRESENTATION_CONTEXT('ca2','external',3);
#840 = CARTESIAN_POINT('', (3.0E+001, 0.0E+000, 1.0E+001));
#850 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#860 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#870 = AXIS2_PLACEMENT_3D('', #840, #850, #860);
/* the link to the external definition in a file has been omitted here
*/

/* #1000 and #1010 establish the relationship between the */
/* geometry of the two shape_aspects                      */
#1000 = (REPRESENTATION_RELATIONSHIP('','',#750,#820)
   REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#1010)
   SHAPE_REPRESENTATION_RELATIONSHIP());
#1010 = ITEM_DEFINED_TRANSFORMATION('aspect_rel','',#795,#870);

/* #1050 to #1140 define the geometric representation of  */
/* the assembly (#150). The shape_representation contains */
/* two axis_placements that are the reference points for  */
/* the transformations of the components                  */
#1050 = PRODUCT_DEFINITION_SHAPE('shape_of_ass',$,#150);
#1060 = SHAPE_DEFINITION_REPRESENTATION(#1050,#1070);
#1070 = SHAPE_REPRESENTATION('s_ass',(#1110,#1140),#1080);
#1080 = GEOMETRIC_REPRESENTATION_CONTEXT('ca','',3);
#1090 = CARTESIAN_POINT('', (3.0E+001, 4.0E+000, 1.0E+001));
#1095 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#1100 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
#1110 = AXIS2_PLACEMENT_3D('', #1090, #1095, #1100);
#1120 = CARTESIAN_POINT('', (1.0E+001, 1.0E+000, 1.0E+001));
#1125 = DIRECTION('', (1.0E+000, 0.0E+000, 0.0E+000));
#1130 = DIRECTION('', (0.0E+000, -1.0E+000, 0.0E+000));
```

```
#1140 = AXIS2_PLACEMENT_3D('', #1120, #1125, #1130);

/* Entities #1200 to #1230 define the geometric relationship */
/* between the component 'part1' and the assembly #150. This */
/* relationship is linked to the product structure relationship */
/* defined by the next_assembly_usage_occurrence #310        */
#1200 = PRODUCT_DEFINITION_SHAPE('shape of assembled p1',$,#310);
#1210 = CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#1220,#1200);
#1220 =(REPRESENTATION_RELATIONSHIP('','',#520,#1070)
   REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#1230)
   SHAPE_REPRESENTATION_RELATIONSHIP());
#1230 =ITEM_DEFINED_TRANSFORMATION('p1t','',#1110,#570);

/* Entities #1300 to #1330 define the geometric relationship */
/* between the component 'part2' and the assembly #150. This */
/* relationship is linked to the product structure relationship */
/* defined by the next_assembly_usage_occurrence #350        */
#1300 = PRODUCT_DEFINITION_SHAPE('shape of assembled p2',$,#350);
#1310 = CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#1320,#1300);
#1320 =(REPRESENTATION_RELATIONSHIP('','',#620,#1070)
   REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#1330)
   SHAPE_REPRESENTATION_RELATIONSHIP());
#1330 =ITEM_DEFINED_TRANSFORMATION('p2t','',#1140,#670);

/* Entities #5000 to #5070 assert that the in #290 defined view */
/* of part2 has a mass of 3KG                                 */
#5000 = GENERAL_PROPERTY('mass property', '', '');
#5010 = GENERAL_PROPERTY_ASSOCIATION('non-definitional', '', #5000,
#5020);
#5020 = PROPERTY_DEFINITION('', '', #290);
#5030 = PROPERTY_DEFINITION_REPRESENTATION(#5020, #5040);
#5040 = REPRESENTATION('property value', (#5060), #5050);
#5050 = REPRESENTATION_CONTEXT('', '');
#5060 = MEASURE_REPRESENTATION_ITEM('mass', MASS_MEASURE(3000), #5070);
#5070 =(NAMED_UNIT(*) MASS_UNIT() SI_UNIT($,.GRAM.));

ENDSEC;
END-ISO-10303-21;
```

Example 22: Complete instantiation for part structure with shape properties

4.5   Other Relationships Between Parts

4.5.1 Alternate Parts

STEP designates alternate and substitute parts differently. Alternate parts are interchangeable in all occurrences whereas substitutes are interchangeable only in a particular usage. Alternate parts in STEP are defined through the alternate_product_relationship entity. This relationship is used in the definition of parts list data for alternate item designations.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of alternate part identification are illustrated in Diagram 27.

primary base part id

part identification
.product #1

base

alternate part id

part identification
.product #2

alternate

alternate _product_relationship

name = ''
definition = ''
basis = 'any use'

Diagram 27 : Alternate Part Instance Diagram

4.5.1.1 alternate_product_relationship

This entity represents the alternate part relationship. The alternate part is interchangeable with the base part in any/all uses - it is a context independent alternate that is form, fit, and function equivalent in any use.

Attributes

- The *name* attribute has no standard mapping in the PDM Schema.
- The *definition* attribute is an optional description.
- The *base* attribute is a reference to the primary product of which an alternate is being identified.
- The *alternate* attribute is a reference to the product identified as an alternate for the base product.
- The *basis* attribute is a rationale for the part interchange (e.g. any use, first available, etc.).

ENTITY alternate_product_relationship	Attribute Population	Remarks
name	type : label = string	
definition	type : text = string	OPTIONAL
base	type : entity = product	
alternate	type : entity = product	
basis	type : text = string	

Preprocessor Recommendations: There are no standard mappings for the name and definition attributes of alternate_product_relationship.   The basis attribute should contain a rationale for the part interchange (e.g. any use, first available, etc.) if it is known.

Postprocessor Recommendations: Since there are no standard mappings for the name and definition attributes of alternate_product_relationship, it is recommended that postprocessors not assign any general processing significance to these values.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context for design assembly life cycle */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');

/* part type discrimination */
#6=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#4, #11));
#7=PRODUCT_CATEGORY_RELATIONSHIP('part-to-detail',$,#6,#8);
#8=PRODUCT_RELATED_PRODUCT_CATEGORY('detail',$,(#4, #11));
```

```
#4=PRODUCT('11111','Solid Box','description for part 11111',(#220));
#11=PRODUCT('11011','Solid Cube','description for part 11011',(#12));

#14=ALTERNATE_PRODUCT_RELATIONSHIP('',$,#11,#4,'interchangeable as
solid squares');
```

Example 23 : exchange file segment for alternate part

4.5.2 Substitute Components in an Assembly

Substitute components in an assembly are identified as a relationship between the occurrences of two part definitions (the two substitutes) that are used within the same parent assembly definition. As opposed to the alternate part relationship, which is context free, this substitute relationship is context dependent - the substitute is only valid in the context of its usage as a component within the specified parent assembly definition.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of substitute part identification are illustrated in Diagram 28.
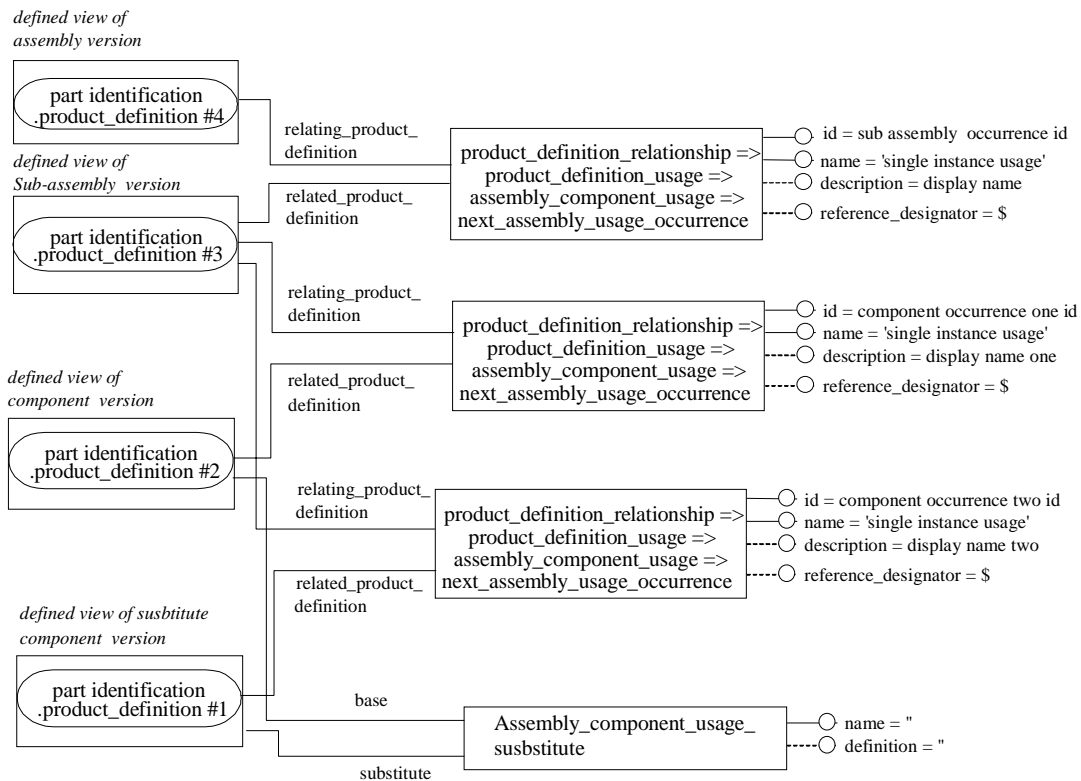
Diagram 28: Substitute Component Instance Diagram

4.5.2.1 assembly_component_usage_substitute

This entity represents the substitute component relationship. The substitute part is interchangeable with the base part only in the particular, specified usage - it is a context dependent substitution that is allowed only in the specific usage as the component in the identified component-assembly relationship.

Attributes

- The *name* attribute has no standard mapping in the PDM Schema.
- The *definition* attribute is an optional description.
- The *base* attribute is a reference to the primary component occurrence of which a substitute is being identified.
- The *substitute* attribute is a reference to the product identified as a substitute for the base component.

ENTITY assembly_component_usage_substitute	Attribute Population	Remarks
name	type : label = string	
definition	type : text = string	OPTIONAL
base	type : entity = assembly_component_usage	
substitute	type : entity = assembly_component_usage	

Preprocessor Recommendations: There are no standard mappings for the name and definition attributes of assembly_component_usage_substitute. The relating_product_definition for both the base and substitute attributes of this relationship must be the same instance representing the same assembly definition.

Postprocessor Recommendations: Since there are no standard mappings for the name and definition attributes of assembly_component_usage_substitute, it is recommended that postprocessors not assign any general processing significance to these values.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#1330 = PROMISSORY_USAGE_OCCURRENCE('seal_in_sleeve',
   'single instance usage',
   'promissory usage of sealing for sleeve assembly ver2', #1210,
#1300, $);
#1335 = PROMISSORY_USAGE_OCCURRENCE('seal_in_sleeve',
   'single instance usage',
   'promissory usage of sealing ver 2 for sleeve assembly ver2', #1210,
#1305, $);
#1337 =
ASSEMBLY_COMPONENT_USAGE_SUBSTITUTE('sealings_alternative_sleeve',
   $,#1330,#1335);
```

Example 24 : exchange file segment for substitute component

4.5.3 Make From Relationships

In STEP, the fact that a part is manufactured from another part is indicated by make_from_usage_option. The make_from_usage_option relates the source part definition in the related_product_definition attribute to the resultant part definition in the relating_product_definition attribute.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of make from part identification are illustrated in Diagram 29.
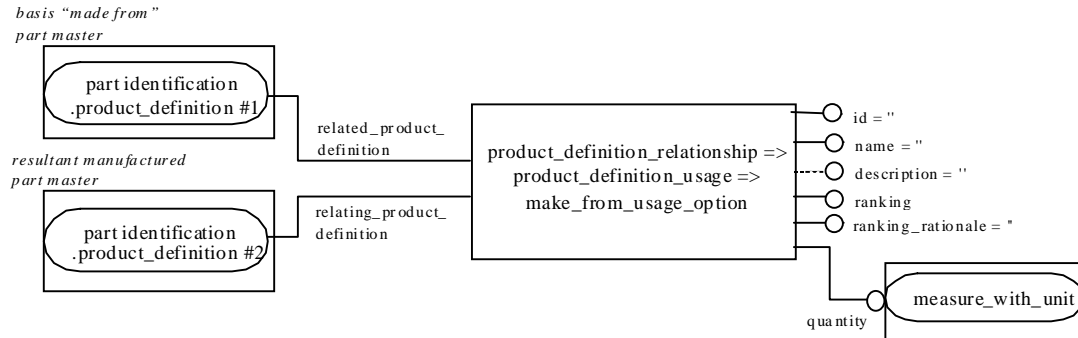
Diagram 29 : Make From Relationship Instance Diagram

4.5.3.1 make_from_usage_option

Entity is a subtype of product_definition_usage. The related_product_definition represents the raw or semi-finished material, and the relating_product_definition represents the resultant manufactured item.

Attributes

- The *relating_product_definition* represents the part manufactured from the related_product_definition.
- The *related_product_definition* represents the raw or semi-finished material.
- The *ranking* attribute is a relative ranking value. A lower value indicates a higher preference.
- The *ranking_rationale* explains the reason for the ranking.
- The *quantity* attribute records the number of resultant parts that can be made from the source part.

ENTITY make_from_usage_option	Attribute Population	Remarks
id	type: identifier = string	unique w.r.t. the relationship
name	type: label = string	
description	type: text = string	OPTIONAL
relating_product_definition	type : entity = product_definition	
related_product_definition	type : entity = product_definition	
ranking	type : integer	
ranking_rationale	type: text = string	
quantity	type: entity = measure_with_unit	

Preprocessor Recommendations: The id attribute must be unique among product_definition_relationships. There is no standard mapping for the name or ranking_rationale attributes in a make_from_usage_option.

Postprocessor Recommendations: Since there is no standard value for the name and ranking_rationale attributes for a make_from_usage_option, it is recommended that postprocessors not assign any processing significance to these values.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context for design assembly life cycle */
#1000 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #1000, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #1000, 'design');

/* part type discrimination */
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#2, #9));

#2=PRODUCT('11000','solid cube','description for part 11000',(#220));
#3=PRODUCT_DEFINITION_FORMATION('A', 'description of version A for
solid cube',#2);
#4=PRODUCT_DEFINITION('D1','detailed drawing as planned for STEP
conformance testing',#3,#230);

#9=PRODUCT('11111','Solid Box','part 11111 made from raw material
11000',(#220));
#10=PRODUCT_DEFINITION_FORMATION('A', 'description of version A for
part 11111',#9);
#11=PRODUCT_DEFINITION('D2','detailed drawing',#10,#230);

#16=MAKE_FROM_USAGE_OPTION('id','name','make two solid boxes from one
solid cube', #11, #4,1,'',#17);
#17 = MEASURE_WITH_UNIT(COUNT_MEASURE(2), #18);
#18 = NAMED_UNIT(#19);
#19 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
```

Example 25 : exchange file for make from relationship

4.5.4 Supplied Part Identification

Supplied part identification is realized by an 'alias relationship' concept. It is specific for supplied parts that allow the renumbering of vendor parts.

In all realms of design and manufacturing business, it is common to buy parts from a vendor and renumber them under an internal numbering scheme. In today's practice, this is done through envelope, specification and source control drawings.

- An envelope drawing is used for a simple renumber of a part where the part is referenced on the envelope drawing and assigned a new part number via the associated parts list.

- A specification control drawing renumbers a part to show that it meets or exceeds the specifications defined on the drawing and to recommend sources for the part.

- A source control drawing renumbers a part and creates a restricted list of suppliers that are qualified to produce the part based on the specifications.

All of these relationships are supported by the product_definition_formation_relationship entity.   This entity is used for the identification of part suppliers. The product_definition_formation_relationship may be used for the renumbering of parts.   The supplied part relationship relates the "new" part master product_definition_formation in the relating_product_definition_formation attribute to the "old" part master product_definition_formation in the related_product_definition_formation attribute.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements of supplied part identification are illustrated in Diagram 30.
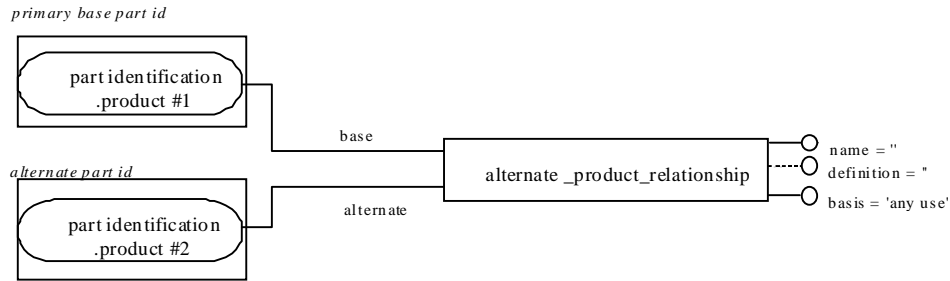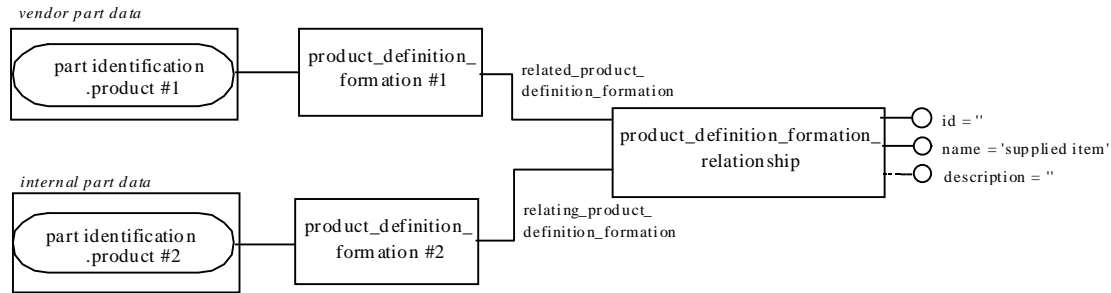

Diagram 30 : Supplied Part Instance Diagram

4.5.4.1 product_definition_formation_relationship

Entity generally relates two product_definition_formation instances in a specified relationship type - the identification of the relationship type is given in the name attribute.

Attributes

- The *id* attribute provides an identifier of the relationship.
- The *name* attribute specifies this relationship represents 'supplied item'.
- The *relating_product_definition_formation* attribute references the internal part version information.
- The *related_product_definition_formation* attribute references the supplied part version information.

ENTITY product_definition_formation_ relationship	Attribute Population	Remarks
id	type: identifier = string	
name	type: label = string	shall be 'supplied item'
description	type: text = string	OPTIONAL
related_product_definition_formation	type : entity = product_definition_formation	supplied part version information
relating_product_definition_formation	type : entity = product_definition_formation	internal part version

Preprocessor Recommendations: The value 'supplied item' for the name attribute identifies this as the supplied item relationship.   There is no standard mapping for the description attribute in a product_definition_formation_relationship. The id attribute must be unique with respect to the relationship, but there is no standard mapping for the value.

Postprocessor Recommendations: Since there are no standard mapping for the description attributes for a product_definition_formation_relationship, it is recommended that postprocessors not assign any processing significance to this value.

Related Entities: Certification of suppliers can be indicated through the supplied item relationship. This is done by relating an applied_certification_assignment to the product_definition_formation_relationship, which associates a certification to the relationship. The applied_certification_assignment entity may have a role associated with it through the entity role_association and its related object_role entity. There are no standard mappings for the values of the role, or the name and purpose attributes of the certification entity.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context for design assembly life cycle */
#1000 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #1000, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #1000, 'design');

/* part type discrimination */
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#2, #9));

/* solid cube - supplied part */
#2=PRODUCT('11000','solid cube','description for part 11000',(#220));
#3=PRODUCT_DEFINITION_FORMATION('A', 'description of version A for
solid cube',#2);
#4=PRODUCT_DEFINITION('D1','detailed drawing as planned for STEP
conformance testing',#3,#230);

/* solid box - internal part */
#9=PRODUCT('11111','Solid Box','part 11111 made from raw material
11000',(#220));
#10=PRODUCT_DEFINITION_FORMATION('A', 'description of version A for
part 11111',#9);
#11=PRODUCT_DEFINITION('D2','detailed drawing',#10,#230);

#16=PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('id','supplied
item',$,#10,#3);
```

Example 26 : exchange file for supplied part

4.5.5 Version History Relationships

There are two types of relationships between product 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 Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of part version sequence history are illustrated in Diagram 31.
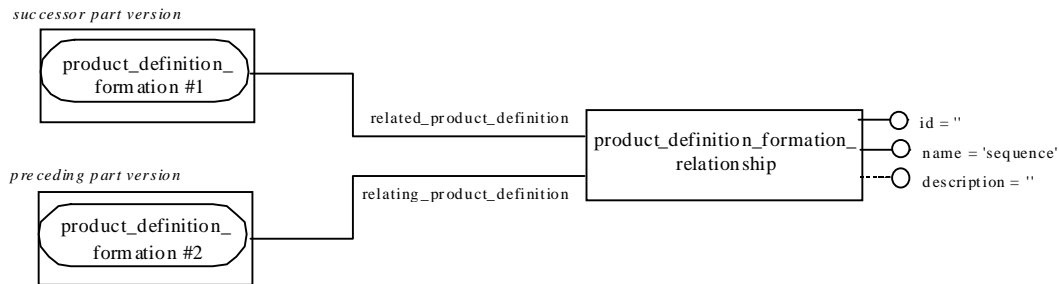
Diagram 31 : Part Version (sequence) History Instance Diagram

4.5.5.1 product_definition_formation_relationship

Entity generally relates two product_definition_formation instances in a specified relationship type - the identification of the relationship type is given in the name attribute.

Attributes

- The *id* attribute provides an identifier of the relationship.

- The *name* attribute specifies this relationship represents a version 'sequence' or 'hierarchy' history.

- The *relating_product_definition_formation* attribute references the preceding (sequence) or super (hierarchy) part version.

- The *related_product_definition_formation* attribute references the subsequent (sequence) or sub (hierarchy) part version.

ENTITY product_definition_formation_relationship	Attribute Population	Remarks
id	type: identifier = string	
name	type: label = string	shall be 'sequence' or 'hierarchy'
description	type: text = string	OPTIONAL
relating_product_definition_formation	type : entity = product_definition_formation	preceding part version
related_product_definition_formation	type : entity = product_definition_formation	subsequent part version

Preprocessor Recommendations: The value 'sequence' for the name attribute identifies this as the version sequence history relationship. The value 'hierarchy' for the name attribute identifies this as the version hierarchical relationship. There is no standard mapping for the description attribute in a product_definition_formation_relationship. The id attribute must be unique with respect to the relationship, but there is no standard mapping for the value.

Postprocessor Recommendations: Since there are no standard mapping for the description attributes for a product_definition_formation_relationship, it is recommended that postprocessors not assign any processing significance to this value.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context for design assembly life cycle */
#1000 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #1000, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #1000, 'design');
/* part type discrimination */
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#2));

/* version A - preceding version */
#2=PRODUCT('11000','solid cube','description for part 11000',(#220));
#3=PRODUCT_DEFINITION_FORMATION('A', 'version A for solid cube',#2);
#4=PRODUCT_DEFINITION('D1','detailed drawing as planned for STEP
conformance testing',#3,#230);

/* version B - subsequent version */
#10=PRODUCT_DEFINITION_FORMATION('B', 'version B for part 11000',#2);
#11=PRODUCT_DEFINITION('D2','detailed drawing',#10,#230);

#16=PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('id','sequence',$,#3,#10)
;
```

Example 27 : exchange file for part version (sequence) history

4.6   Complete instantiation example for part structure and relationships

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('Assembly instantiation example'), '2;1');
FILE_NAME('', '11.06.1999, 08:29:45', (''), (''), '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;

/* Entities #10 to #520 define that items with associated meta-   */
/* information that are referenced in the product structure       */
/* definitions below                                              */
#10 = APPLICATION_CONTEXT('');
#20 = PRODUCT_CONTEXT('', #10, '');
#30 = PRODUCT('dca', 'disc cap assembly',
   'disc cap assembly in dissassembly view of hub assembly', (#20));
#40 = PRODUCT('h1', 'hub assembly', '', (#20));
#50 = PRODUCT('s1', 'sleeve assembly', '', (#20));
#60 = PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#40, #50, #30));
#70 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#40, #50, #30));
#80 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #70, #60);
#90 = PRODUCT('seal1', 'sealing', '', (#20));
#100 = PRODUCT('d1', 'disc', 'disc with holes', (#20));
#110 = PRODUCT('c1', 'cap', 'cap for hub assembly', (#20));
#120 = PRODUCT('cy1', 'cylinder', 'cyclinder for sleeve assembly',
(#20));
#130 = PRODUCT('g1', 'gasket', 'gasket for sleeve assembly', (#20));
#140 = PRODUCT('gex', 'gasket extern', 'externally supplied gasket', (
   #20));
#150 = PRODUCT('mat23', 'raw material for caps', '', (#20));
#160 = PRODUCT('cl_alt', 'cylinder alternative',
```

```
     'alternate product for cy1', (#20));
#170 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#40, #100, #110,
    #50, #120, #130, #30, #140, #150, #160, #90));
#180 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema',
1998
    , #10);
#190 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#100, #110, #120,
    #130, #140, #160, #90));
#200 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#100, #110, #120,
    #130, #140, #160, #90));
#210 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #200, #190);
#220 = PRODUCT_DEFINITION_CONTEXT_ROLE('', $);
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, '');
#240 = PRODUCT_DEFINITION_FORMATION('next_ass',
    'hub assembly version 2 (with individual occurrences of components',
#40);
#250 = PRODUCT_DEFINITION('view 1 hub', 'design view on hub assembly',
    #240, #230);
#260 = APPLICATION_CONTEXT('mechanical design');
#270 = PRODUCT_DEFINITION_CONTEXT('', #260, 'design');
#280 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#250, #270, #220);
#290 = PRODUCT_DEFINITION_FORMATION('1', '', #100);
#300 = PRODUCT_DEFINITION('disc view 1', 'design view on disc', #290,
    #230);
#310 = APPLICATION_CONTEXT('mechnical design');
#320 = PRODUCT_DEFINITION_CONTEXT('', #310, 'design');
#330 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#300, #320, #220);
#340 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
#360 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#250, #350, #340);
#370 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('disc1', 'single instance usage',
'', #250, #300, $);
#390 = PRODUCT_DEFINITION_FORMATION('1', '', #130);
#400 = PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);
#410 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#400, #270, #220);
#420 = PRODUCT_DEFINITION_FORMATION('1', '', #120);
#430 = PRODUCT_DEFINITION('cv1', 'design view on cylinder', #420,
#230);
#440 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#430, #270, #220);
#450 = PRODUCT_DEFINITION_FORMATION('1', '', #50);
#460 = PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly', #450
    , #230);
#470 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #270, #220);
#480 = PRODUCT_DEFINITION_FORMATION('1', '', #110);
#490 = PRODUCT_DEFINITION('capv1', 'design view on cap', #480, #230);
#500 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#490, #270, #220);
#520 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);

/* Instances #530 to #570 define the product structure for      */
/* the hub assembly of #250 via the comepoents including the     */
/* sub-assembly of the sleeve #460                               */
#530 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu1', 'single instance usage',
    'gasket usage 1', #460, #400, $);
#540 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu2', 'single instance usage',
    'gasket usage 2', #460, #400, $);
#550 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('cyl1', 'single instance usage',
    'cyclinder usage', #460, #430, $);
```

```
#560 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('cap1', 'single instance usage',
   'usage of cap in hub assembly', #250, #490, $);
#570 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('svu1', 'single instance usage',
   'usage of sleeve sub-assembly in hub assembly', #250, #460, $);

/* Entites #580 to #700 define the 'id owners' for the parts  */
#580 = ORGANIZATION('com1', 'Xample Company', 'company');
#590 = ORGANIZATION_ROLE('id_owner');
#600 = APPLIED_ORGANIZATION_ASSIGNMENT(#580, #590, (#40));
#610 = ORGANIZATION_ROLE('id owner');
#620 = APPLIED_ORGANIZATION_ASSIGNMENT(#580, #610, (#130));
#630 = ORGANIZATION_ROLE('id owner');
#640 = APPLIED_ORGANIZATION_ASSIGNMENT(#580, #630, (#120));
#650 = ORGANIZATION_ROLE('id owner');
#660 = APPLIED_ORGANIZATION_ASSIGNMENT(#580, #650, (#50));
#670 = ORGANIZATION_ROLE('id owner');
#680 = APPLIED_ORGANIZATION_ASSIGNMENT(#580, #670, (#110));
#690 = ORGANIZATION_ROLE('id owner');
#700 = APPLIED_ORGANIZATION_ASSIGNMENT(#580, #690, (#100));


/* Entity #710 is an alternate view on the hub assembly version   */
/* defined by the product_definition_formation #240               */
/* The product structure in this view is defined by the entities  */
/* #760 to #810                                                    */
/* This product structure is different to the view #240 of the    */
/* same part version defined by #250. In this example the alternate */
/* product structure related to the view is a dissassembly structure */
/* as opposed to the assembly structure related to the view #250    */
#710 = PRODUCT_DEFINITION('view 2 hub', 'dissassembly view on hub',
#240, #230);
#720 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#710, $, #220);
#730 = PRODUCT_DEFINITION_FORMATION('1', '', #30);
#740 = PRODUCT_DEFINITION('dcav1', 'dissassembly view on disc and cap',
   #730, #230);
#750 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#740, $, #220);
#760 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('du', 'single instance usage',
   'disc in disc cap assembly', #740, #300, $);
#770 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#740, #350, #340);
#780 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('cu', 'single instance usage',
   'cap in disc-cap assembly', #740, #490, $);
#790 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#710, #350, #340);
#800 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('su2', 'single instance usage',
   'sleeve assembly in dissassembly view of hub', #710, #460, $);
#810 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('dcau1', 'single instance usage',
'disc cap assembly usage in dissassembly view of hub', #710, #740, $);


/* #820 is another version of the hub assembly defined by #40      */
/* This version is specified with a product structure that         */
/* quantifies the occurrences of components in an assembly         */
/* The product structure relationships that defines the qunaitifed */
/* usage of the components in an assembly are instaniated complex   */
/* with NEXT_ASSEMBLY_COMPONENT_USAGE to indicate that a given      */
/* parent node directly uses the referenced component              */
#820 = PRODUCT_DEFINITION_FORMATION('quantif',
   'version specified with qunantified  usages', #40);
#830 = PRODUCT_DEFINITION('qv1',
```

```
   'view on quantified version of hub assembly', #820, #230);
#840 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#830, $, #220);
#850 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#830, #350, #340);
#860 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
#870 = NAMED_UNIT(#860);
#880 = MEASURE_WITH_UNIT(COUNT_MEASURE(1), #870);
#890 = (ASSEMBLY_COMPONENT_USAGE($) NEXT_ASSEMBLY_USAGE_OCCURRENCE()
   PRODUCT_DEFINITION_RELATIONSHIP('du4', 'quantified instance usage',
   'quantified usage of disc', #830, #300) PRODUCT_DEFINITION_USAGE()
   QUANTIFIED_ASSEMBLY_COMPONENT_USAGE(#880));
#900 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
#910 = NAMED_UNIT(#900);
#920 = MEASURE_WITH_UNIT(COUNT_MEASURE(1), #910);
#930 = (ASSEMBLY_COMPONENT_USAGE($) NEXT_ASSEMBLY_USAGE_OCCURRENCE()
   PRODUCT_DEFINITION_RELATIONSHIP('cau4', 'quantified instance usage',
'quanitifed usage if cap', #830, #490) PRODUCT_DEFINITION_USAGE()
   QUANTIFIED_ASSEMBLY_COMPONENT_USAGE(#920));
#940 = PRODUCT_DEFINITION_FORMATION('slq',
   'quantifed version of sleeve assembly', #50);
#950 = PRODUCT_DEFINITION('slqv',
   'design view quantified version sleeve assembly', #940, #230);
#960 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#950, $, #220);
#970 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#950, #350, #340);
#980 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
#990 = NAMED_UNIT(#980);
#1000 = MEASURE_WITH_UNIT(COUNT_MEASURE(1), #990);
#1010 = (ASSEMBLY_COMPONENT_USAGE($) NEXT_ASSEMBLY_USAGE_OCCURRENCE()
   PRODUCT_DEFINITION_RELATIONSHIP('cu4', 'quantified instance usage',
   'qunatifed usage of cylinder', #950, #430)
PRODUCT_DEFINITION_USAGE()
    QUANTIFIED_ASSEMBLY_COMPONENT_USAGE(#1000));
#1020 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
#1030 = NAMED_UNIT(#1020);
#1040 = MEASURE_WITH_UNIT(COUNT_MEASURE(2), #1030);
#1050 = (ASSEMBLY_COMPONENT_USAGE($) NEXT_ASSEMBLY_USAGE_OCCURRENCE()
   PRODUCT_DEFINITION_RELATIONSHIP('gu4', 'quantified instance usage',
   'quantifed usage of gasket for sleeve', #950, #400)
   PRODUCT_DEFINITION_USAGE()
QUANTIFIED_ASSEMBLY_COMPONENT_USAGE(#1040));

/* #140 represents an externally supplied gasket part          */
/* The relationship #1070 relates the version 1 of this externally */
/* supplied part to the internal part master represented by      */
/* #140 in its version 1 (defined by #390) */
#1060 = PRODUCT_DEFINITION_FORMATION('1', '', #140);
#1070 = PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('', 'supplied item',
   'supplied item version of gasket', #1060, #390);
#1080 = ORGANIZATION('com2', 'Supplier Company', 'company');
#1090 = ORGANIZATION_ROLE('id owner');
#1100 = APPLIED_ORGANIZATION_ASSIGNMENT(#1080, #1090, (#1060));

/* product #140 represents the raw material for the cap product  */
/* (#110), version 1 (#480) in the design view (#490) */
```

```
#1120 = PRODUCT_RELATED_PRODUCT_CATEGORY('raw material', '', (#150));
#1140 = PRODUCT_DEFINITION_FORMATION('1', '', #150);
#1150 = PRODUCT_DEFINITION('mat_view', 'design view on mat23', #1140,
   #230);
#1160 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#1150, $, #220);
#1170 = ORGANIZATION_ROLE('id owner');
#1180 = APPLIED_ORGANIZATION_ASSIGNMENT(#580, #1170, (#1140));


/* Entities #1181 to #1185 establish the relation that defines that  */
/* the cap (in view of #490) is made from the raw material specified */
/* via #1150 and related entities                                    */
#1181 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
#1182 = NAMED_UNIT(#1181);
#1184 = MEASURE_WITH_UNIT(COUNT_MEASURE(1), #1182);
#1185 = MAKE_FROM_USAGE_OPTION('mkfr1','','',#490,#1150,1,'',#1184);



/* #1190 models the successor version of the sleeve assembly    */
/* version 1 define by #450. 'Sequence' indicates that #1190 is */
/* a direct successor to #450                                   */
#1190 = PRODUCT_DEFINITION_FORMATION('2',
   'successor version of version 1', #50);
#1200 = PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('', 'sequence', '',
   #450, #1190);


/* #1210 specifies a view on the sleeve assembly version 2. To this  */
/* view a product structure is attached that extends the product     */
/* structure of version one with an additonal sealing part           */
/* For this sealing no detail is given, a                            */
/* promissory_usage_occurrence is used to model the intent to */
/* include the sealing in the sleeve assembly                 */
#1210 = PRODUCT_DEFINITION('sv2', 'design view on sleeve version 2',
   #1190, #230);
#1220 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#1210, $, #220);
#1230 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#1210, #350, #340);
#1240 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu5', 'single instance usage',
   'gasket usage for ver2 of sleeve assembly', #1210, #400, $);
#1250 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu6', 'single instance usage',
   '2ns gasket usage for sleeve assembly ver2', #1210, #400, $);
#1260 = PRODUCT_DEFINITION_FORMATION('1', '', #160);
#1270 = PRODUCT_DEFINITION('v1_cyl_alt',
   'design view on alternate cylinder', #1260, #230);
#1280 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#1270, $, #220);
#1290 = PRODUCT_DEFINITION_FORMATION('1', 'sealing ver1', #90);
#1295 = PRODUCT_DEFINITION_FORMATION('2', 'sealing ver2', #90);
#1300 = PRODUCT_DEFINITION('sv1', 'design view on sealing v1', #1290,
#230);
#1305 = PRODUCT_DEFINITION('sv2', 'design view on sealing v2', #1295,
#230);
#1310 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#1300, $, #220);
#1320 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('cylalt', 'single instance
usage'
   , 'usage of alternate cylcinder ofr sleeve v2', #1210, #1270, $);

/* #1330 models the intent to use the sealing part as defined */
/* by #1300 in the sleeve part as defined by #1210            */
```

```
#1330 = PROMISSORY_USAGE_OCCURRENCE('seal_in_sleeve',
   'single instance usage',
   'promissory usage of sealing for sleeve assembly ver2', #1210,
#1300,
   $);


/* The entities #1335 and #1337 define a usage of another version of */
/* the sealing part given by its view in #1305 in the sleeve         */
/* assembly #1210. #1337 specifies that the usage of the two version */
/* of the sealing in the sleeve assembly is alternative, i.e. the    */
/* product structure relationship defined by #1335 is alternative to */
/* the one defined by #1330                                          */
#1335 = PROMISSORY_USAGE_OCCURRENCE('seal_in_sleeve',
   'single instance usage',
   'promissory usage of sealing ver 2 for sleeve assembly ver2', #1210,
#1305,
   $);
#1337 =
ASSEMBLY_COMPONENT_USAGE_SUBSTITUTE('sealings_alternative_sleeve',
   $,#1330,#1335);


/* #1340 is a version of the hub assembly #40 that is derived from */
/* the version #240 of the same part. The derivation relationship is */
/* established by entity #1350                                      */
#1340 = PRODUCT_DEFINITION_FORMATION('ver_double',
   'double sleeve version', #40);
#1350 = PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('', 'derivation', '',
   #240, #1340);
#1360 = PRODUCT_DEFINITION('dsl_view',
   'design view on double sleeve version of hub', #1340, #230);
#1370 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#1360, $, #220);
#1380 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#1360, #350, #340);


/* The assembly structure below assembles the #1340 version of the */
/* hub assembly with 2 sleeves. the sleeves each consist of two */
/* gaskets and and one cylinder. Thus in the resulting product */
/* structure there are four cylinders. The basic next_assembly_usage */
/* construct allows only to distinguish between two gaskets one the */
/* level of the sleeve assembly definition. To indivdiulize one of */
/* the gaskets in the hub assembly specified_higer_usage_occurrence */
/* is used.                        */
#1390 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('dcau', 'single instance usage',
'usage of dca assembly for double sleeve version', #1360, #740, $);
#1400 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('usl1', 'single instance usage',
'1st sleeve in double sleeve version', #1360, #460, $);
#1410 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('usl2', 'single instance usage',
'2nd sleeve in double sleeve assembly', #1360, #460, $);


/* #1415 indvidualizes the first gasket (#530) used in the        */
/*  second sleeve (#1410) of the assembly #1360.                  */
#1415 = SPECIFIED_HIGHER_USAGE_OCCURRENCE('2nd_g',
  'identification of second gasket of first sleeve',
  $,
  #1360,    /* the hub assembly  */
  #400,     /* gasket used for the assembly */
  $,        /* reference_designator        */
  #1410,    /* upper_usage: select the second sleeve usage */
```

```
  #530      /* next_usage_: select first usage of gasket in sleeve */
  );

/* #1420 specifies that the alternate cylinder part #160 is an     */
/* alternative to the cylinder #120. These parts are mutually      */
/* exchangeable in any context                                     */
#1420 =
ALTERNATE_PRODUCT_RELATIONSHIP('hub_alternativity',$,#160,#120,'text
for basis');
ENDSEC;
END-ISO-10303-21;
```

Example 28 : exchange file for complete part structure and relationships

4.7 Known issues

4.7.1 Version Hierarchy Relationship

Version history information has been described as the sequence relationship between part versions (see 4.5.5). The entity product_definition_formation_relationship with attribute name value 'sequence' represents this information. In addition, the value 'hierarchy' is used for this name attribute, to identify a hierarchy relationship between two versions of different levels, e.g. a version and an iteration. This representation of iterations as well as released versions (revisions) of a part is not yet completely understood in relation to the sequence history or the implementation of current PDM systems.

4.7.2 Item Find Number

Legacy AP203 implementations have used the name attribute of the next_assembly_usage_occurrence entity to represent the find_number (or position_number) of a part as it appears on a physical drawing. This conflicts with the AP214 usage of this attribute to distinguish single instances from selected instances.

5 Document Identification

The PDM Schema deals with documents as products, according to a basic STEP interpretation of 'Document as Product'. As with 'Part as Product', there are three basic concepts central to document identification in the PDM Schema:

- product master identification,
- context information,
- type classification.

These fundamental concepts are described for 'Part as Product' in this document. The following provides specifics on the distinction and additions for the 'Document as Product' approach.

'Document as Product' identifies a managed document object in a PDM system. A managed document is under revision control, and may distinguish various representation definitions of a document version. The document_version represents the minimum identification of a managed document under revision control. A document representation definition may optionally be associated with one or more constituent external files that make it up.

EXAMPLE - a configuration controlled managed paper document like a drawing would generally map to a usage of the entity 'product', with version identification and a (physical) representation/view definition according to the 'document as product' approach.

External files in the PDM Schema represent a simple external reference to a named file. An external file is not managed independently by the system - there is usually no revision control or any representation definitions of external files. Version identification may optionally be associated with an external file, but this is for information only and is not used for managed revision control.

If a file is under configuration control, it should be represented as a constituent of a document definition view/representation. 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 mapped as a constituent of a view/representation definition of the new document version. A simple external reference alone is not configuration controlled, it is just an external file reference to product data.

Documents may be associated with product data in a specified role, to represent some relationship between a document and other elements of product data. Constraints may also be specified on this association, in order to distinguish an applicable portion of an entire document or file in the association. With 'Document as Product', additional entities are required to relate a managed document to other product data (see Section 10.1). Included among these is the entity document. The document.id should not be used as valid user data - the document entity does not always need to be instantiated using 'Document as Product', it is done only to assign the document to other product data via applied_document_reference.

5.1 Document as Product

The interpretation of 'Document as Product' uses basic product master identification for the fundamental requirements of document identification, versioning, and representation definition. 'Document as Product' is distinguished from 'Part as Product' in each of the three basic elements of product identification:

- product master identification - document identification has specific requirements to assign documents to other product data, and to optionally associate with the constituent external file(s) that make up a specific document representation view definition;
- context information - document identification has different context information than part identification;
- type classification - document identification has a different type classification than part identification.

5.1.1 Document Master Identification

As with 'Part as Product', the concepts of base identification, version identification, and view definition are structurally distinct in 'Document as Product'. The general recommendations given for part master identification apply to the document master identification, except where differences are noted.

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.

With 'Document as Product' the product view definition is used to define a view of a particular representation of a document version. A document version does not have to have an associated document representation definition.

The representation view definition of a document version is used for association of document properties, to build document structures, or to associate a document with the set of constituent external files that make it up.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support identification of a document version are illustrated in Diagram 32. Document version identification without an associated document view representation definition represents the minimum requirements for document master identification.
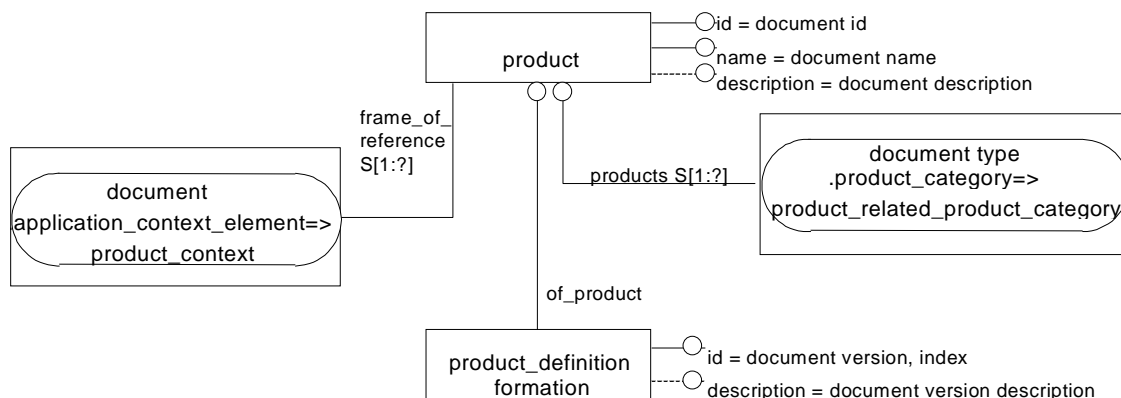
Diagram 32: Minimum Document Identification Instance Diagram

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#270=PRODUCT('doc_4711','packaging guide','packaging guideline for
part',(#260));
#280=PRODUCT_DEFINITION_FORMATION('ver3.1','version 3.1',#270);
```

Example 29: exchange file segment for minimum document identification

5.1.2 Context Information

Context information provides a scope and necessary circumstance for product identification information. It consists of two separate and related areas:

- Application Protocol Identification,

- Application Context Information.

Application protocol identification and general context information are handled structurally the same for 'Document as Product', 'Part as Product', and product concept identification. A single instance of the entity application_context should be referenced by all product_context and product_concept_context entities, for parts, documents and product concepts. This application_context should be referenced by the single instance of the entity application_protocol_definition. The general context information identifies the usage of the information within the scope of the PDM Schema and the application domain which provides a basis for the interpretation of all information represented in the product data exchange.

The application context information is managed differently to distinguish documents from parts:

- Life-cycle information does not have to be maintained for 'Document as Product', as it is for 'Part as Product' to identify the development life-cycle stages. When a managed document is assigned to a part master, the life-cycle stage of the part master identification may be extrapolated as relevant to the document;

- To distinguish a document representation view definition, the names 'digital document definition' or 'physical document definition' are used, as differentiated from the value 'part definition' used in 'Part as Product'. A digital document definition represents an electronic document, while a physical document definition stands for a 'hardcopy', typically paper, document.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of document identification with context information are illustrated in Diagram 33.
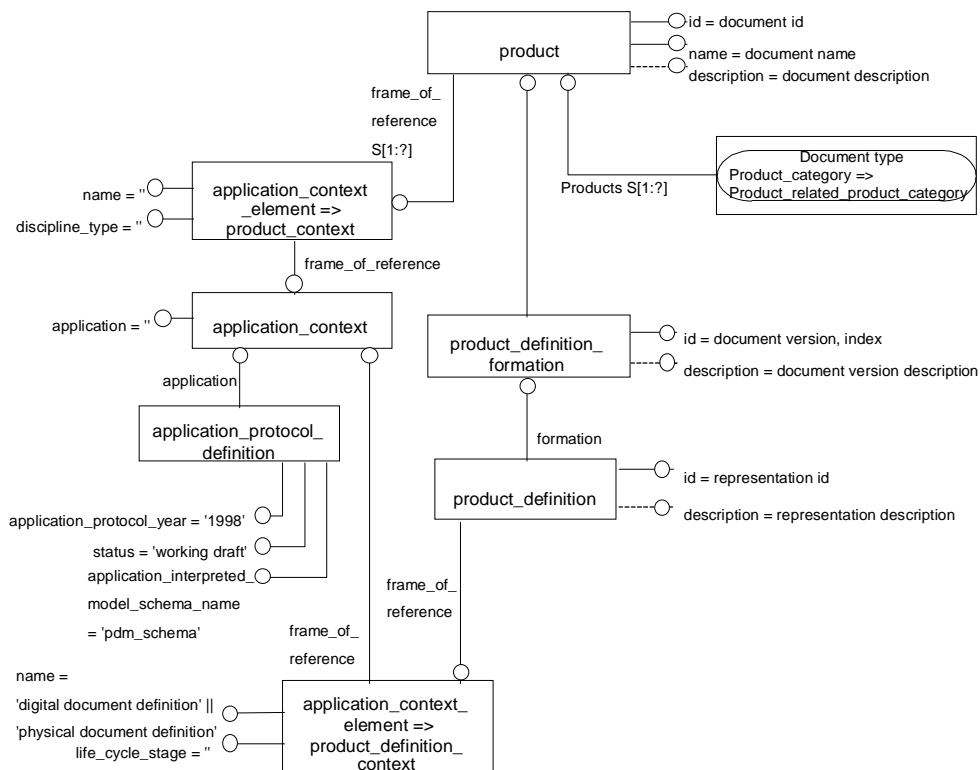
Diagram 33: Document Master with Context Information Instance Diagram

5.1.2.1 product_definition

The product_definition entity denotes the definition of a particular view of a representation of a document version. There may be more than one document representation definition associated with a single document version. The representation view definition of a document version is used for association of document properties, to build document structures, or to associate a document with the set of constituent external files that make it up. The entity product_definition supports property association and document structure. The subtype product_definition_with_associated_documents is used to associate a representation of a document version with the set of constituent files that make it up.

Attributes
- The *id* attribute indicates the unique identifier of the document representation definition.
- The *description* attribute provides optional additional words describing the representation definition.
- The *formation* attribute references the document version of which this is a representation definition.
- The *frame_of_reference* attribute references the context information related to this definition.

ENTITY product_definition	Attribute Population	Remarks
id	type : identifier = string	Must be unique in relation with a specific product_version
description	type: text	OPTIONAL
formation	type: entity = product_definition_formation	Document version of which this is a particular document representation definition
frame_of_reference	type : entity = product_definition_context	Context information for this definition.

Preprocessor Recommendations: The value for the id attribute of product_definition should be unique relative to other product_definition entities related to the same product_definition_formation.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

5.1.2.2 product_definition_context

All STEP product_definitions must be defined in a product_definition_context. With 'Document as Product' the context information is used to distinguish digital from physical 'hardcopy' documents.

Attributes
- The *name* attribute indicates the type of the document representation definition.
- The *frame_of_reference* attribute references application domain information.
- The *life_cycle_stage* attribute has no standard mapping for documents in this usage guide.

ENTITY product_definition_context	Attribute Population	Remarks
name	type: label = string 'digital document definition' or 'physical document definition'	Distinguishes the associated product_definition as that of a document
frame_of_reference	type: entity = application_context	
life_cycle_stage	type: label = text	

Preprocessor Recommendations: The name attribute distinguishes the representation definition of a document version as either digital ('digital document definition') or physical, i.e., hardcopy ('physical document definition').

Postprocessor Recommendations: Postprocessors should interpret the value of the name attribute as a distinction between part and document definitions, and further between digital and physical document definitions.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#240=APPLICATION_CONTEXT('');
#245=APPLICATION_PROTOCOL_DEFINITION('version
1.1','pdm_schema',1998,#240);
#250=PRODUCT_DEFINITION_CONTEXT('digital document definition',#240,'');
#260=PRODUCT_CONTEXT('',#240,'');
#270=PRODUCT('doc_4711','packaging guide','packaging guideline for
part',(#260));
#280=PRODUCT_DEFINITION_FORMATION('ver3.1','version 3.1',#270);
#290=PRODUCT_DEFINITION('id of digital document','digital document for
representing guideline ver3.1',#280,#250);
```

Example 30: exchange file segment for document master with context information

5.1.3 Type Classification

Type classification information provides the basic capability to distinguish products interpreted to represent documents from those interpreted as parts.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the complete requirements of document identification with context information and type classification are illustrated in Diagram 34.
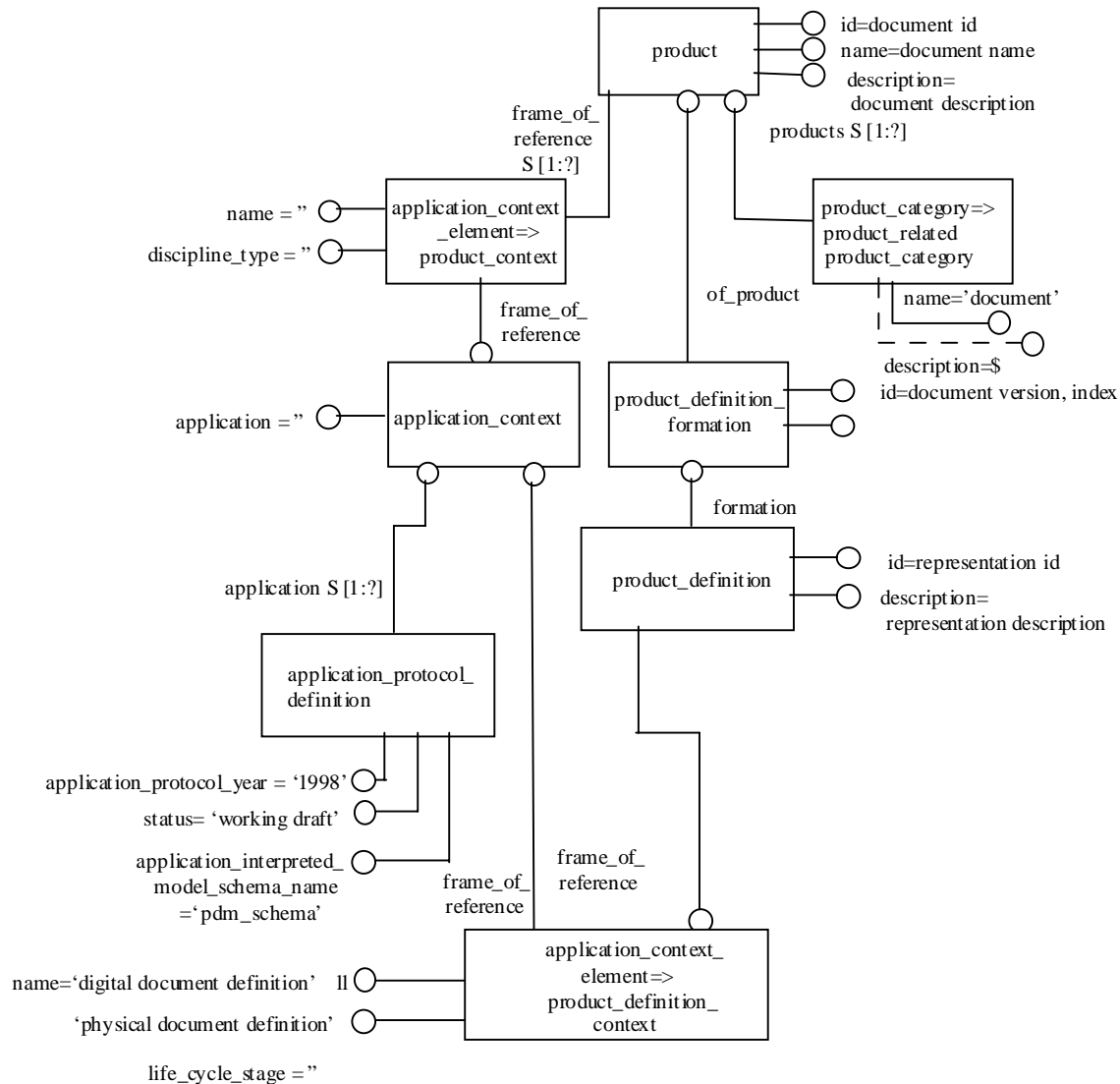
Diagram 34: Complete Document Master with Context and Type Classification Instance Diagram

5.1.3.1 product_related_product_category

The product_related_product_category represents the identification of a specific classification applied to a product. The name and description attributes are inherited from the supertype product_category. This subtype adds the attribute products that allow it to be associated (related) directly to a product instance. In the usage scenario of 'Document as Product', these product instances are used to represent managed documents.

Attributes
- The *products* attribute associates the category with the document as product to which it applies.

ENTITY product_related_ product_category	Attribute Population	Remarks
name	type: label = string 'document'	

ENTITY product_related_ product_category	Attribute Population	Remarks
description	type: text = string	
products	type: entity = product	SET[1:?] of product entity instance(s) distinguished as representing documents.

Preprocessor Recommendations: The name attribute should have the value 'document' to discriminate 'Document as Product' from 'Part as Product'.

Postprocessor Recommendations: When the value of the attribute name is 'document', postprocessors should recognize that the value(s) of the attribute products are the representation of managed documents as opposed to parts.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#240=APPLICATION_CONTEXT('');
#245=APPLICATION_PROTOCOL_DEFINITION('version
1.1','pdm_schema',1998,#240);
#250=PRODUCT_DEFINITION_CONTEXT('digital document definition',#240,'');
#260=PRODUCT_CONTEXT('',#240,'');
#270=PRODUCT('doc_4711','packaging guide','packaging guideline for
part',(#260));
#280=PRODUCT_DEFINITION_FORMATION('ver3.1','version 3.1',#270);
#290=PRODUCT_DEFINITION('id of digital document','digital document for
representing guideline ver3.1',#280,#250);
#300=PRODUCT_RELATED_PRODUCT_CATEGORY('document',$,(#270));
```

Example 31: exchange file for complete document master with context and type classification

6   Specific Document Type Classification

The PDM Schema supports the specific classification of both parts and managed documents. Consequentially following the 'document as product' approach as described in this usage guide, the type classification of managed documents is done in symmetry to the specific type classification for parts.

There are two classification mechanisms that can be used when exchanging classification information with STEP. One type of classification works by assigning categories to product data items. These categories are identified by name labels that define the related classification. This type of classification is referred to as specific classification.

NOTE - As an advanced requirement there might be the need to classify product data items according to a classification system with explicit reference to the classification criteria and related properties of the product data items. This classification mechanism is called general classification. The PDM Schema only supports specific classification of product data items via assigned categories, which are defined by labeling them with a name.

All the categories used for the specific document type classification of managed documents should be defined in a single taxonomy. The category hierarchy is defined via relating the categories as super/sub-categories.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of specific document type classification are illustrated in Diagram 35.
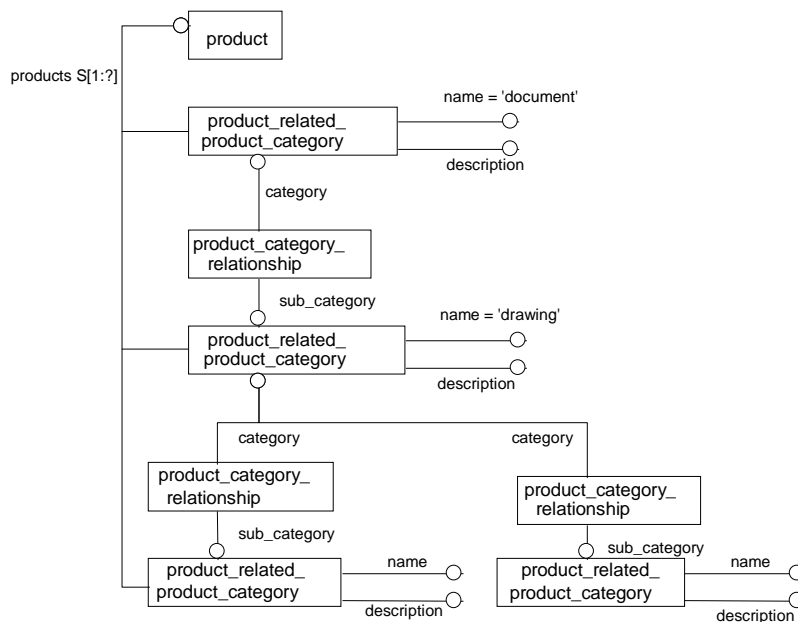
Diagram 35: Specific document classification Instance Diagram

6.1.1.1 product_related_product_category

The product_related_product_category entity is a subtype of product_category.

Attributes
- The *name* attribute identifies the category via a label.
- The *description* provides additional characterization for the product category.
- The *products* attribute associates the category with the product entity instances to which it applies.

ENTITY product_related_ product_category	Attribute Population	Remarks
name	type: label = text	
description	type: text = string	OPTIONAL
products	type: entity = product	SET[1:?]

Preprocessor Recommendations: For reasons of robustness (see postprocessor recommendations below) preprocessors should list all products in the attribute product_related_product_category.items that are either direct members of that category or which belong to sub-categories of this category. A given product category shall only be instantiated once in an exchange file.

The specific classification capability relies exclusively on the name labels of the categories for the interpretation of the category semantics. Exchanges of specific classification information are only meaningful when the category names are agreed upon between the exchange partners. For the PDM Schema, it is recommended that the highest level category be instantiated with the value 'document'. Exchange partners may extend it to create a hierarchy based on bilateral agreements.

Postprocessor Recommendations: Postprocessors should also tolerate instances of product_category as opposed to product_related_product_category in the intermediate levels of a product category hierarchy definition. At least processors should be robust enough to ignore the product_category instances. If these instances are ignored then the detailed information of the product category hierarchy is not completely transferred, nevertheless the assignment of products to categories is still maintained since these assignments always use product_related_product_categories.

Related Entities: The entity product_category is a supertype of product_related_product_category that does not have the attribute products. Thus a product_category can only be used an intermediate node in a classification hierarchy to which no products are directly assigned.

6.1.1.2 product_category_relationship

The product_category_relationship entity establishes a category-subcategory relationship between two product_category entities.

Attributes
- The *name* attribute gives the word or group of words by which the relationship is referred.
- The *description* attribute provides additional descriptive information related about the relationship.
- The *category* attribute identifies the *super* category (more general) in the relationship.
- The *sub_category* attribute identifies the *sub* category (more specific) in the relationship.

ENTITY product_category_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	
category	type: entity = product_related_product_category	
sub_category	type: entity = product_related_product_category	

Preprocessor Recommendations: The PDM Schema does not have restrictions on potential nesting of category hierarchies besides that it is strongly recommended that the structure is acyclic. Nevertheless preprocessing systems should be aware that not all postprocessing systems might support an arbitrary deep structuring of hierarchy levels.

Postprocessor Recommendations: Postprocessors should at least support a two level category hierarchy. This is in alignment with the requirement to process the basic product instance type classification, which is to be done on two levels for product entity instances that model parts being assemblies or details.

Related Entities: Instances of product_category_relationship relate instances of product_category and product_related_product_category.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('specifc doc classification example'),
    '2;1');
FILE_NAME('', '10.09.1999, 13:16:45', ('n.n.'), (''),
    '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
#10 = PRODUCT_DEFINITION_FORMATION('1', '', #20);
#20 = PRODUCT('d1', 'drawing_x', '', (#40));
#30 = PRODUCT_RELATED_PRODUCT_CATEGORY('document', $, (#20));
#40 = PRODUCT_CONTEXT('', #50, '');
#50 = APPLICATION_CONTEXT('');
#60 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema', 1998
    , #50);
#120 = PRODUCT_CATEGORY_RELATIONSHIP('',$,#30,#70);
#70 = PRODUCT_RELATED_PRODUCT_CATEGORY('drawing', $, (#20));
#80 = PRODUCT_CATEGORY_RELATIONSHIP('',$,#70,#90);
#90 = PRODUCT_RELATED_PRODUCT_CATEGORY('technical drawing', $, (#20));
#100 = PRODUCT_CATEGORY_RELATIONSHIP('',$,#90,#110);
#110 = PRODUCT_RELATED_PRODUCT_CATEGORY('detail drawing', $, (#20));
/* other product data follows here ... */
ENDSEC;
END-ISO-10303-21;
```

Example 32 : exchange file for specific document classification

6.2   Known issues

6.2.1   Availability of specific document classification capability in AP 214 DIS

specific_item_classification is only applied to items (e.g. parts) in the AP214 DIS ARM. This capability should also be applied to documents that are realized by the 'document as product' approach. This is appropriate for the type classification of a document. A comment has been issued against the AP214 DIS.

7   External Files

External files in the PDM Schema represent a simple external reference to a named file. The external file may identify a digital file or a physical, 'hardcopy' file. As opposed to a managed 'Document as Product', 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. In the case where properties differ with different versions, the managed 'Document as Product' approach is recommended.

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 mapped as a constituent of a view/representation definition of the new document version. A simple external reference alone is not configuration controlled, it is just an external file reference to product data.

While managed revision control representing multiple versions and version history is not available for external files, external files may have an optional version identification providing a string labeling the version of the file.

7.1   External File Identification

Identification of an external file is done using the entity document_file. The document_file entity is a defined subtype of the entity document. It is also a subtype of the entity characterized_object, which allows association of properties to the identification of an external file.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of external file identification are illustrated in Diagram 36 below.
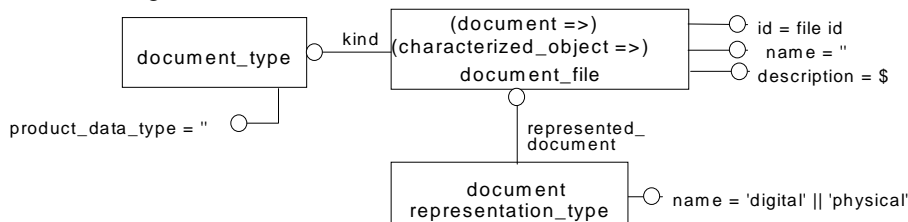
Diagram 36: External File Identification Instance Diagram

7.1.1.1 document_file

This entity is a subtype of the document entity, and thus works together with the applied_document_- reference to support the assignment of external files to product data. Document_file is also a subtype of the entity characterized_object, which has local attributes name and description.

Since both supertypes of this entity define local attributes 'name' and 'description', the subtype document_file has a double inheritance of these attributes. Local constraints defined on the entity document_file specify that the additional attributes inherited from the supertype characterized_object should not be used for valid user data. These constraints currently require that the value of these attributes be a string with a single space: ' '. This is an identified error in the EXPRESS, it should specify a value of

the empty string ". This EXPRESS error will be corrected in the future.   It is recommended that implementations use the empty string ".

Attributes
- The *id* attribute indicates the unique identifier of the external file.
- The *name* attribute is double inherited. Only the one derived from the document supertype is valid for user data, the nomenclature of the external file.   The name attribute inherited from characterized_object should be assigned the empty string.
- The *description* attribute is double inherited. Only the one derived from the supertype document is valid for user data. The description attribute inherited from characterized_object should be assigned the empty string.
- The *kind* attribute is a reference to the file "type" classification information.

ENTITY document_file	Attribute Population	Remarks
id	type: identifier = string	
SELF/document.name	type: label = string	valid user data for file id
SELF/document.description	type: text = string	OPTIONAL
kind	type: entity = document_type	
SELF/characterized_object.name	type: label = string empty string "	not to be used for valid user data
SELF/characterized_object.description	type: text = string empty string "	not to be used for valid user data

Preprocessor Recommendations: Preprocessors must carefully encode the exchange syntax for an instance of document_file to properly handle the multiple inheritance of the attributes name and description. An example instance of document_file is illustrated in Example 33.

The document_file entity requires an associated instance of the entity document_representation_type. The document_file entity constrains the possible values of the attribute document_representation_type.name, as described in 7.1.1.2.

The id attribute is a unique identifier for the external file. Access path information for a file should be represented as a file "source" property.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

7.1.1.2 document_representation_type

This entity provides the capability to identify the type of the representation of a particular external file, either digital or physical.

Attributes
- The *name* attribute is used to identify the particular representation type (either digital or physical) of the associated external file.
- The *represented_document* attribute is a reference to the associated external file.

ENTITY document_representation_type	Attribute Population	Remarks
name	type: text = string 'digital' or 'physical'	
represented_document	type: entity = document	for file identification, this attribute will reference the subtype document_file

Preprocessor Recommendations: For file_identification, the possible values for the name attribute are 'digital' and 'physical'. A digital file represents an electronic file on a computer system. A physical file is the actual paper hardcopy or other physical realization of a file.

Postprocessor Recommendations: A value for the name attribute of 'digital' should be interpreted to mean the associated document_file represents an external reference to an electronic digital file. The value 'physical' should be interpreted to mean the associated document_file represents an external reference to a hardcopy file.

Related Entities: There are no specific related entities.

7.1.1.3 document_type

This entity provides the capability to identify the type of an external file for the general requirement of file type classification.

Attributes

* The *product_data_type* attribute is used to identify the kind of product data in the associated file.

ENTITY document_type	Attribute Population	Remarks
product_data_type	type: identifier = string	

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#510=DOCUMENT_FILE('doc_id','',$,#520,'','');
#520=DOCUMENT_TYPE('unspecified type');
#540=DOCUMENT_REPRESENTATION_TYPE('digital',#510);
```

Example 33: exchange file segment for external file identification

Although managed revision control representing multiple versions and version history is not available for external files, they may have an optional version identification that provides a string labeling the version of the external file. If multiple versions are represented, the 'Document as Product' approach is recommended.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements of external files with optional version identification are illustrated in Diagram 37 below.
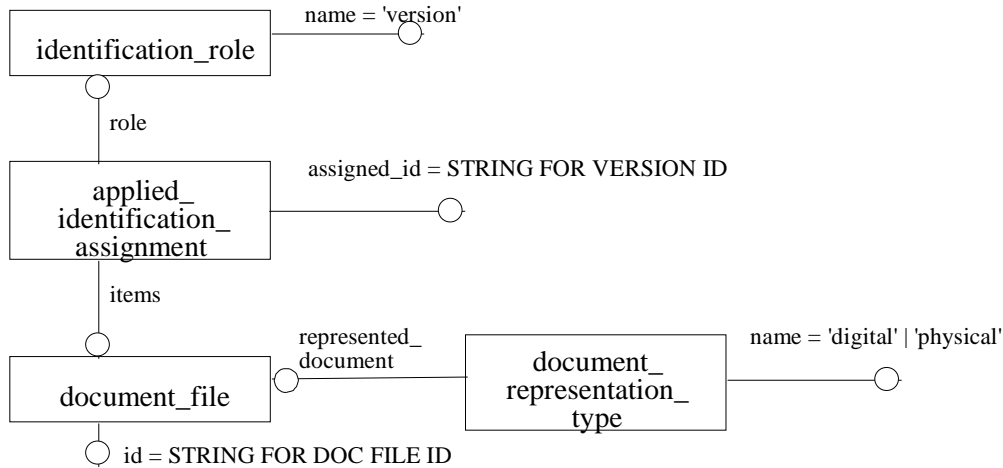
Diagram 37: External File with Version Identification Instance Diagram

7.1.1.4 applied_identification_assignment

This entity is a subtype of the entity identification_assignment. It allows the actual assignment of an identification_assignment entity to product data, in this case to a document_file entity representing an external file.

Attributes

- The *items* attribute is used to reference the associated external file.

ENTITY applied_identification_assignment	Attribute Population	Remarks
assigned_id	type: identifier = string	contains the string identifying the version of the external file.
role	type: entity = identification_role	for file version identification, this entity should have name attribute value = 'version'
items	type : entity = document_file	reference to the associated external file to which the version identification is assigned.

Preprocessor Recommendations: The use of this entity is optional. When applied to a document_file, it represents simple version identification for the external file. It is not to be used for managed revision control. If managed revision control is the requirement, then the 'Document as Product' approach must be used. The value of the attribute assigned_id contains the version identification. This should not be used for managed version control, but rather for an optional label providing additional information about the version of the document that is identified. The value of the role attribute is an instance of the entity identification_role with its name attribute assigned the string 'version'.

Postprocessor Recommendations: Postprocessors should recognize this entity as providing version identification label for the associated external file.

Related Entities: There are no specific related entities.

7.1.1.5 identification_role

This entity in this use case indicates the role of a version identification for an external file.

Attributes

- The *name* attribute is used to indicate that the related identification_assignment represents a version identification for the associated external file.
- The *description* attribute is optional additional text.

ENTITY identification_role	Attribute Population	Remarks
name	type: identifier = string 'version'	Contains the string indicating that the related identification_assignment represents a version identification for an external file.
description	type: text = string	OPTIONAL

Preprocessor Recommendations: The attribute name should be assigned a value of 'version' in this usage.

Postprocessor Recommendations: The name attribute value 'version' should be interpreted as an indication that the associated applied_identification_assignment represents simple version identification for the file.

*Related Entities***:** There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#500=IDENTIFICATION_ROLE('version',$);
#510=DOCUMENT_FILE('doc id','',$,#520,'','');
#520=DOCUMENT_TYPE('unspecified type');
#530=APPLIED_IDENTIFICATION_ASSIGNMENT('THE VERSION ID',#500,(#510));
#540=DOCUMENT_REPRESENTATION_TYPE('digital',#510);
```

Example 34: exchange file for external file with version identification

8 Relationship Between Documents and Constituent Files

In 'Document as Product', the view definition is used to represent a definition of a particular document representation. There may be more than one representation definition associated with a document version. The document representation definition may be associated with the constituent external files that make it up. The association of constituent files with the definition of a document representation is optional.

If a file is under configuration control, it should be represented as a constituent of a document definition view/representation. 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 mapped as a constituent of a view/representation definition of the new document version. A simple external reference alone is not configuration controlled, it is just an external file reference to product data.

The AIM entity product_definition_with_associated_documents is used together with the identification of external files to support this requirement.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements of document identification with associated constituent external files are illustrated in Diagram 38.
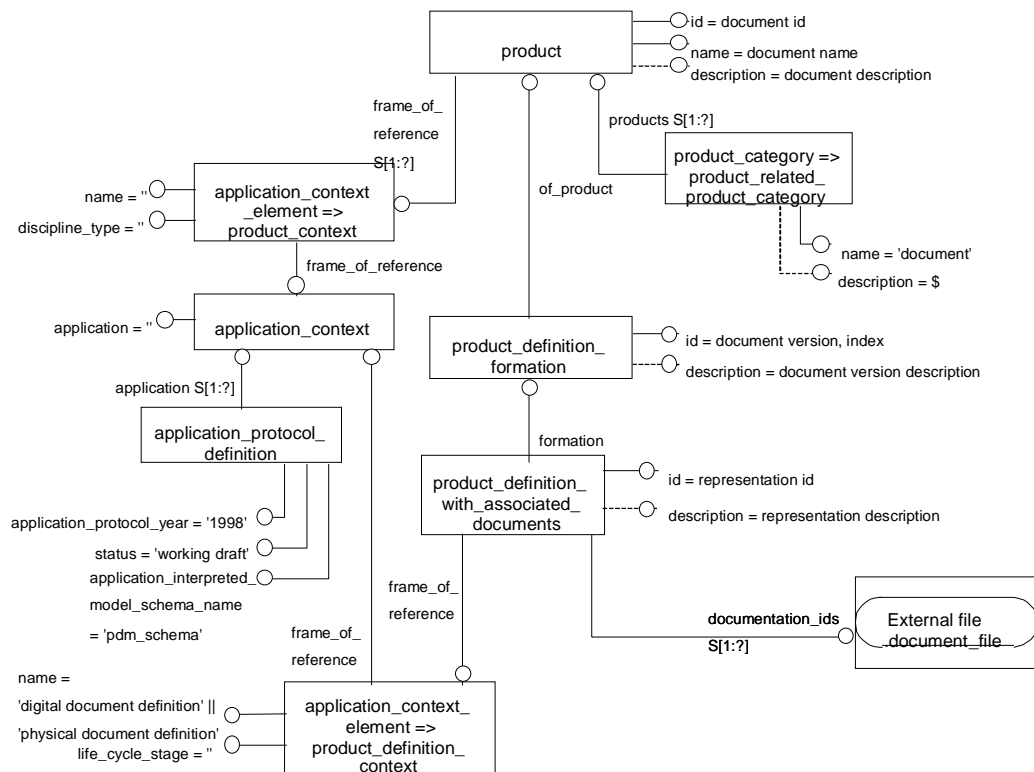
Diagram 38: Document Master with Constituent External Files Instance Diagram

8.1.1.1 product_definition_with_associated_documents

The product_definition_with_associated_documents entity is a subtype of product_definition. It inherits the attributes id, description, formation and frame_of_reference from the supertype. This entity is only used in this case of 'Document as Product' and is not to be used in the case of 'Part as Product'. The attribute documentation_ids provides the relationship to the external file(s) that make up the actual content of the document representation definition.

Attributes

- The ***documentation_ids*** attribute contains a set of at least one instance representing the external file(s) that compose this view definition of the document.

ENTITY product_definition_with_ associated_documents	Attribute Population	Remarks
id	type: identifier = string	the identifier of this document view definition, should be unique in relation to a specific version
description	type: text = string	optional
formation	type: entity = product_definition_formation	References associated product_definition_formation
frame_of_reference	type: entity = product_definition_context	reference to the associated product_definition_context
documentation_ids	type: entity = document_file	SET[1:?] of files that make up the definition of the document

Preprocessor Recommendations: Preprocessors should use this entity instead of the supertype product_definition when relating the representation view definition of a 'Document as Product' to its constituent external files. The value of the attribute id should be unique in relation to a specific version.

Postprocessor Recommendations: Postprocessors should interpret this entity as a document view representation definition that identifies the constituent external files that make it up.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#270=PRODUCT('doc_4711','packaging guide','packaging guideline for
part',(#260));
#300=PRODUCT_RELATED_PRODUCT_CATEGORY('document',$,(#270));
#310=DOCUMENT_TYPE('unspecified');
#320=DOCUMENT_FILE('t1','text.doc','file with text for the
guide',#310,'','');
#330=PRODUCT_DEFINITION_FORMATION('ver3.2','version 3.2',#270);
#340=DOCUMENT_FILE('l1','',$,#310,'','');
#440=DOCUMENT_REPRESENTATION_TYPE('digital',#340);
#350=PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS('rep_id_3.2','represe
ntation of version 3.2',#330,#250,(#340,#320));
```

Example 35: exchange file for document with constituent external files

9   Document and File Properties

Document properties can in the PDM Schema be associated to representations of documents as well as to individual files. If document properties are assigned to representations of documents the characteristics apply as well to all the constituent files of the document representation in most cases. Some properties with numeric values, such as 'file size' and 'page count', applied to the document representation will not correspond to an individual file in a multiple file document representation, but to the sum of all the files that make up the particular document representation definition. To avoid redundancy it is recommended that properties that are shared by all constituents of a given document representation are directly associated with that document representation rather than replicated with the individual files.

The Instance Model: EXPRESS entities and attributes

The general schema for the assignment of properties to document representation (i.e. product_definition) or document files (i.e. document_file) is to instantiate a property definition tree with property_definition, property_definition_representation, representation and eventually multiple instantiations of descriptive_representation_item (or measure_representation_item) that provide the representation of a specific aspect of the represented property.
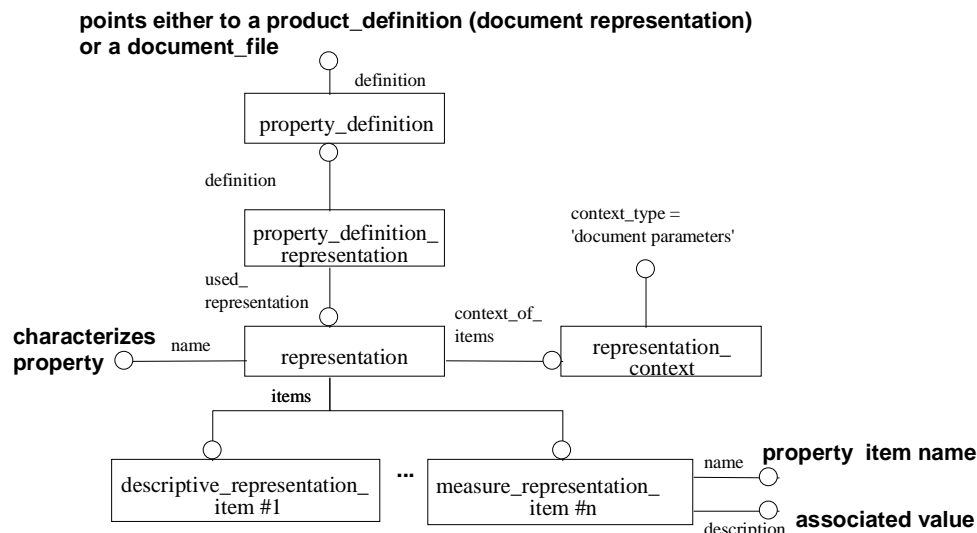
Diagram 39: General pattern for the association of document properties

9.1.1.1 property_definition

A property_definition is in the given context a property that characterizes a document representation or document file. It applies either to a product_definition (for the document representation) or a document_file.

Attributes

- The *name* attribute indicates via 'document property' that the property is related to a document representation or a document file.
- The *description* attribute provides additional description of the property.
- The *definition* attribute references the document object, which is characterized by the property_definition.

ENTITY property_definition	Attribute Population	Remarks
name	type: identifier = string	should be instantiated as 'document property' in the given context
description	type: text = string	optional, shall not be instantiated
definition	type: entity = product_definition or product_definition_with_ associated_documents or document_file	References associated document

Preprocessor Recommendations: It is recommended to instantiate not more than one property_definition with the value of the name attribute equal to 'document property' for each document representation definition (product_definition or product_definition_with_associated_documents) or document_file. One property_definition per document object shall be used to collect all document object properties via associated property_definition_representations.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

9.1.1.2 property_definition_representation

A property_definition_representation is in the given context an association between a document property and its representation.

Attributes

- The ***definition*** attribute references the property_definition that is defined by the associated representation.
- The ***used_representation*** attribute points to a representation that collects the representations items that together describe the property.

ENTITY property_definition_representation	Attribute Population	Remarks
definition	type: entity = property_definition	References associated property_definition
used_representation	type: entity = representation	References associated representation

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

9.1.1.3 representation

A representation is in the context of document properties a collection of one or more descriptive_representations_items that are related in a representation_context with type 'document parameters'. Herein a representation represents a document property.

Attributes

- The ***name*** attribute characterizes the type of the document related property via a string.
- The ***items*** attribute collect items that represent the values for a given property instantiation.

- The *context_of_items* attribute points to a representation_context. The representation_context has the type 'document parameters'.

ENTITY representation	Attribute Population	Remarks
name	type: label = STRING	the name attribute characterizes the document related property
items	type: entity = descriptive_representation_item	the items that constitute the representation of the document property
context_of_items	type: entity = representation_context	is required to point to a representation_context with representation_context.type set to 'document parameters'
id	derived	should not be instantiated
description	derived	should not be instantiated

Preprocessor Recommendations: The name characterizes the document related property. The representation is required to have a context with representation_context.type = 'document parameters'. For a given document representation or document_file, the name of all associated representation objects shall be different for the properties described in sections 1.2 through 1.5.

Postprocessor Recommendations: A postprocessor shall in the given context support the following values for representation.name 'document content', 'document creation', 'document format', 'document size'.

Related Entities: None specific.

9.1.1.4 representation_context

A representation_context is identifies the context (of type 'document parameters') of interpretation for the value of items in a representation.

Attributes
- The *context_identifier* attribute identifies the context.
- The *context_type* attribute specifies the type of the context.

ENTITY representation_context	Attribute Population	Remarks
context_identifier	type: label = STRING	
context_type	type: text = STRING	should be instantiated with value 'document parameters'.

Preprocessor Recommendations: In order to distinguish the use of a representation for document properties, the context_type attribute of the representation_context entity has the value 'document parameters'.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

9.1.1.5 descriptive_representation_item

A descriptive_representation_item is in the document property context a textual element that participates in one or more representations to define the respective properties.

Attributes

- The *name* attribute characterizes the information modeled with the descriptive_representation_item via a string.
- The *description* attribute defines a textual value as an instantiation of the modeled property.

ENTITY descriptive_representation_item	Attribute Population	Remarks
name	type: label = STRING	the name attribute indicates the name of the represented property
description	type: text = STRING	the description is the value associated with the representation item in textual form

Preprocessor Recommendations: None specific

Postprocessor Recommendations: In the given context the following instantiations of descriptive_representation_item.name shall be expected: 'detail level', 'geometry', 'language', 'real world scale', 'creating interface', 'creating system', 'operating system', 'data format', 'character code', 'size format', 'size format standard'.

Related Entities: The descriptive_representation_items for a given document property are collected in a representation. The representation characterizes the property via the attribute representation.name.

9.2   Document content property

The content related properties of documents are represented accordingly to the general scheme outlined above.   To indicate that the property is content related the used representation must be instantiated with:

representation.name = 'document content'

This document content capability specifies characteristics detailing the content of a given document object. Aspects of the content property can be modeled via the following instantiations of descriptive_representation_items collected in the above representation.

requirement	descriptive_- representation _item.name	descriptive_representation_item.description
The level of detail that the document file or the document representation provides.	'detail level'	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.
The kind or kinds of geometry that an object contains	'geometry	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 overlayed 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.
language or languages are used in the characterized objects.	'language'	e.g. 'English''
the scale that is used	'real world scale'	e.g. '1:50'

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Entities #210 to #250 assign document content parameters to 'file1'
(#80)
#210 = PROPERTY_DEFINITION('document property', '', #80);
#220 = PROPERTY_DEFINITION_REPRESENTATION(#210, #230);
#230 = REPRESENTATION('document content', (#240, #250), #200);
#240 = DESCRIPTIVE_REPRESENTATION_ITEM('detail level', 'rough 3D
shape');
#250 = DESCRIPTIVE_REPRESENTATION_ITEM('geometry type', 'solid model');
```

Example 36: exchange file segment for content property representation

9.3   Document creation property

The content related properties of documents are represented accordingly to the general scheme outlined above. To indicate that the property is related to document creation the used representation must be instantiated with:

representation.name = 'document creation'

Aspects of the content property can be modeled via the following instantiations of descriptive_representation_items collected in the above representation. It is recommended that when document creation property information is represented in a corresponding representation then at least a descriptive_representation_item with name 'creating system' shall be instantiated.

requirement	descriptive_-representation_item.name	descriptive_representation_item.description
the computer application used to create the document object.	'creating interface"	e.g. Postscript driver"
the computer application or the machine which is used to create the object that is characterized.	'creating system'	e.g. 'Microsoft Word V6'
the operating system that is used to execute the computer application that created the characterized object.	'operating system'	e.g. 'HP-UX 11'

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Entities #210 to #250 assign document creation parameters to 'file1'
(#80)*/
#260 = PROPERTY_DEFINITION('document property', '', #80);
#270 = PROPERTY_DEFINITION_REPRESENTATION(#260, #280);
#280 = REPRESENTATION('document creation', (#290, #300, #310), #200);
#290 = DESCRIPTIVE_REPRESENTATION_ITEM('creating system', 'My CAD');
#300 = DESCRIPTIVE_REPRESENTATION_ITEM('operating system', 'Linux
2.1');
#310 = DESCRIPTIVE_REPRESENTATION_ITEM('creating interface',
   'export driver');
```

Example 37: exchange file segment for creation property representation

9.4   Document format property

The format related properties of documents are represented accordingly to the general scheme outlined above. To indicate that the property is related to document format the used representation must be instantiated with:

representation.name = 'document format'

This capability specifies the format of a document object. At least one of the items shall be specified for each instance of this property representation. Aspects of the format property can be modeled via the following instantiations of descriptive_representation_items collected in the above representation.

requirement	descriptive_-representation_item.name	descriptive_representation_item.description
the convention	'data format'	where applicable the following values shall be used:

that was used to structure the information in the characterized object		'DXF', 'IGES', 'STEP AP203', 'STEP AP214', 'TIFF CCITT GR4', 'VDAFS', 'VOXEL'
the character code that is used for the stored data	'character code'	where applicable the following values shall be used: 'US ASCII 7bit', 'ISO LATIN-1', 'EBCDIC', 'binary'
the dimensions of a physical presentation	'size format' or 'size format standard' where applicable	e.g. 'ISO A0', '0.2 x 0.4 x 0.4 meters'

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Entities #140 to #200 assign document format parameters to 'file1'
(#80)
#140 = PROPERTY_DEFINITION('document property', '', #80);
#150 = PROPERTY_DEFINITION_REPRESENTATION(#140, #160);
#160 = REPRESENTATION('document format', (#170, #190), #200);
#170 = DESCRIPTIVE_REPRESENTATION_ITEM('data format', 'VOXEL');
#190 = DESCRIPTIVE_REPRESENTATION_ITEM('character code', 'binary');
#200 = REPRESENTATION_CONTEXT('', 'document parameters');
```

Example 38: exchange file segment for format property representation

9.5   Document size property

The document size related properties of documents are represented accordingly to the general scheme outlined above. To indicate that the property is content related the used representation must be instantiated with:

representation.name = 'document size'

This capability specifies the size of document object. At least one of the items shall be specified for each instance of this property representation. The size property capability differs from the general scheme in the fact that measure_representation_items are instantiated instead of descriptive_representation_items. Aspects of the size property can be modeled via the following instantiations of measure_representation_items collected in the representation.

requirement	measure_representation_item.name
the value that represents the size of a digitally stored document.	'file size'
the number of pages (of a physical document)	'page count'

9.5.1.1 measure_representation_item

A measure_representation_item is in the document property context an element that participates in one or more representations to define the respective properties by defining measure values with associated units.

Attributes

- The *name* attribute characterizes the information modeled with the measure_representation_item via a string.
- The *value_component* attribute defines a value as an instantiation of the modeled property.
- The *unit_component* attribute provides the unit for the size or page counts measure.

ENTITY measure_representation_item	Attribute Population	Remarks
name	type: label = STRING	must be 'file size' or 'page count' in the given context
value_component	type: select measure_value = REAL | -NUMBER	to indicate the select type it is recommended to instaniate COUNT_MEASURE(nr) for page numbers and CONTEXT_DEPENDENT_MEASURE(size) for 'file size'
unit_component	type: select unit = NAMED_UNIT | DERIVED_UNIT	It is recommended to use a context_dependent_unit as unit component

Preprocessor Recommendations: For 'page count' it is recommended to instantiate an INTEGER as value_component. For 'file size' it is recommended to instantiate a REAL. The related dimensional_exponents should all be set to 0.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#90 = DOCUMENT_TYPE('');
#100 = DOCUMENT_FILE('hardcopy', '', $, #90, '', '');
#110 = DOCUMENT_REPRESENTATION_TYPE('physical', #100);
#120 = PROPERTY_DEFINITION('document property', '', #100);
#130 = PROPERTY_DEFINITION_REPRESENTATION(#120, #140);
#140 = REPRESENTATION('document size', (#150), #200);
#150 = MEASURE_REPRESENTATION_ITEM('page count', COUNT_MEASURE(1),
#170);
#160 = DIMENSIONAL_EXPONENTS(0.0E+000, 0.0E+000, 0.0E+000, 0.0E+000,
   0.0E+000, 0.0E+000, 0.0E+000);
#170 = CONTEXT_DEPENDENT_UNIT('pages',#160);
```

Example 39: exchange file segment for size property representation

9.6   Document source property

This capability allows the specification of location of a document object in a digital or physical data storage system. This capability differs in its instantiation from the general document property pattern.

A central idea in capturing this requirement is that the document location comprises two components: storage node identification and path information. For digital files, for example, the storage item identification is equivalent to the file name, whereas the path information might describe the Internet node and directory in which the file can be found. The storage item identification is captured in applied_external_identification_assignment.assigned_id and the path information is mapped onto external_source.source_id.

The attribute identification_role.name is used to specify the mechanism being applied to identify the storage item and path information. Typical values for this attribute are 'URL' or ISBN in the case of physical documents.

While applied_external_identification_assignment.assigned_id captures the file name, the attributes of the entity document_file allow for the specification of a unique identifier (attribute id) and nomenclature associated with the document or file (attribute name). The identifier is not necessarily the file name.

However, if no document source property is assigned to a document_file, the following semantics may apply, depending on business process agreements:

- Document_file.id may specify the file name;
- In this case, document_file.id would suffice to unambiguously identify the document_file. In particular, it means that file names are defined in the context of a local directory.

The attribute identification_role.description specifies whether the location is associated with a source system, destination system or with some node form which the file or document can be retrieved. For example, the name of a given file used in a technical data package assembled for exchange purposes may differ from the name for the equivalent file in the source system.

The Instance Model: EXPRESS entities and attributes

The instantiation diagram is shown below in Diagram 40.
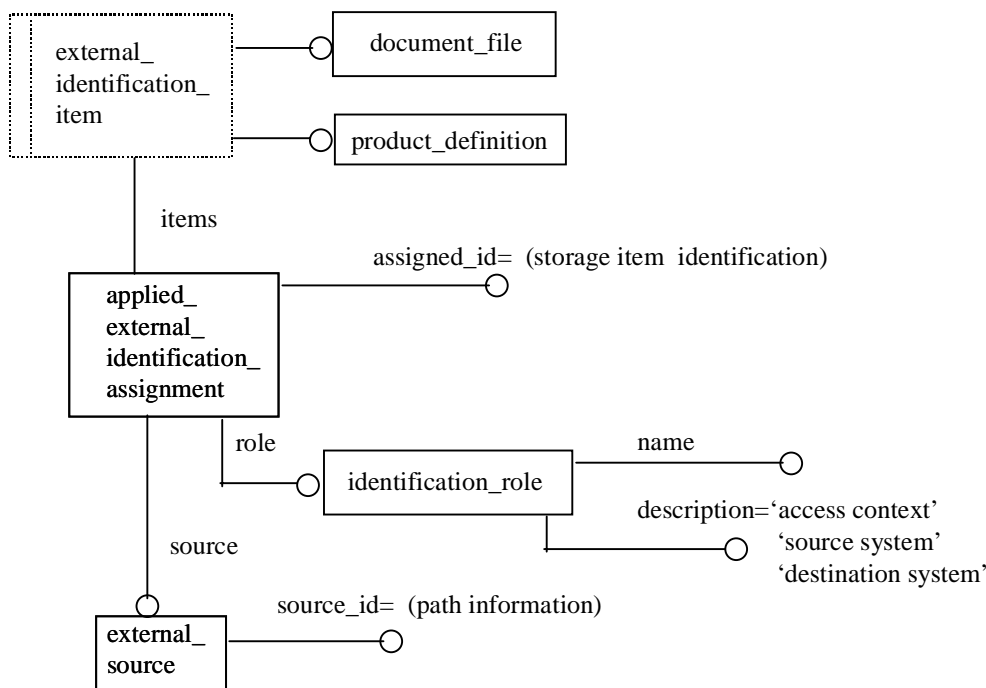
Diagram 40: Document source property instance diagram

9.6.1.1 identification_role

Identification role provides a name and a description for an identification assignment. In the context of this capability this entity specifies that the assignment structure be used to represent the document source property.

Attributes
- The *name* attribute characterizes the mechanism used to identify the storage item and the path information associated with the file or the representation of a document. Recommended values are:

- ‘URL’,
- ‘FTP’,
- ‘ISBN’ – for physical documents,
- ‘Technical Data Package’ – meaning that the location is associated with a technical data package being used for data exchange purposes,
- ‘tracking’ – for physical models.
- The *description* attribute identifies the context associated with the location of the file or document.

ENTITY identification_role	Attribute Population	Remarks
name	type: label = STRING	
description	type: text = STRING	Must be ‘source system’, ‘destination system’ or ‘access context’

Preprocessor Recommendations: Identification_role.description shall be instantiated as 'source system', ‘destination system’ or ‘access context'' in the given context.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

9.6.1.2 applied_external_identification_assignment

Applied_external_identification_assignment is used to assign an external source and identifier to a set of items.

Attributes
- The *assigned_id* attribute captures the storage node identification.
- The *role* attribute specifies the identification_role instance, which captures the location context and the mechanism used to express the location.
- The *items* attribute is a set of document representations or document files for which the document source information is valid.

ENTITY applied_external_identification_-assignment	Attribute Population	Remarks
Assigned_id	type: label = STRING	
Role	type: entity = IDENTIFICATION_ROLE	
Items	type: set of entity = product_definition (for document representations) or document_file	document objects for which the document source property applies

Preprocessor Recommendations: Applied_external_identification_assignment.assigned_id shall be instantiated with meaningful storage node location information of the file, document representation or physical module, even if this information is duplicated in the document_file attributes id or name.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

9.6.1.3 external_source

An external_source is the identification of a source of product related data. In the context of the document source property the external source provides the name of the source by which it can be accessed in a storage system.

Attributes

- The *source_id* attribute defines the path information associated with the file or document location.

ENTITY external_source	Attribute Population	Remarks
source_id	type source_item: STRING	specifies the path information
Description	type text: STRING	DERIVED: Instantiation of this attribute is not required.

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

9.7   Additional Document Properties

Several implementations and pilot projects have identified the requirement to exchange additional properties beyond the pre-defined values given above. One example of such an additional defined property is that of notation related to a document, originating in the AP232.

9.7.1 Document Notation

General notation associated with documents are represented accordingly to the general scheme outlined above. To indicate that the property is related to document notation the used representation must be instantiated with:

<div align="center">representation.name = 'notation'</div>

This capability specifies explanatory remarks or notations related to a document object that may be useful or imformative to a human. An identifier may be associated with each note using the attribute descriptive_representation_item.name.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Entities #140 to #200 assign document format parameters to 'file1'
(#80)
#140 = PROPERTY_DEFINITION('document property', '', #80);
#150 = PROPERTY_DEFINITION_REPRESENTATION(#140, #160);
#160 = REPRESENTATION('notation', (#170), #200);
#170 = DESCRIPTIVE_REPRESENTATION_ITEM('rc1', 'actual notation text
identified by the reference code rc1');
#200 = REPRESENTATION_CONTEXT('', 'document parameters');
```

<div align="center">**Example 40 : exchange file segment for document notation property**</div>

9.8   Document type classification

The PDM Schema supports the assignment of document type information to documents, specific representations of documents, and document files. Type classification of documents is discussed in Sections 5.1.3 and 6. The other approaches are outlined in the following subsections. In the case where a document type property is assigned to the representation of a document, the type characteristics apply to all

individual file objects that are the constituents of the document representation. In the case of an assignment to a document file, the type characteristics apply on an individual basis. The PDM Schema does not support the assignment of a classification system relative to which the classification has been done.

9.8.1 Document type classification for document files

A document_file provides in its attribute *kind* a pointer to the entity document_type. Document_type.product_data_type can be used for the classification of the type of the document_file.

9.8.1.1 document_type

Attributes

- The ***product_data_type*** attribute describes the type of product data represented by the document entity.

ENTITY document_type	Attribute Population	Remarks
product_data_type	type: identifier = string	

Preprocessor Recommendations: Where applicable the following values shall be used:
- 'geometry': The document file represents a shape model;
- 'NC data': The document file represents numerical control data;
- 'FE data': The document file represents finite element data;
- 'sample data': The document file represents measured data;
- 'process plan': The document file represents process planning data;
- 'check plan': The document file represents quality control planning data;
- 'drawing': The document file represents a technical drawing.

Postprocessor Recommendations: None specific.

Related Entities: This entity is required for each instance of the entity document.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#80 = DOCUMENT_FILE('file1', '', $, #90, '', '');
#90 = DOCUMENT_TYPE('geometry');
#100 = DOCUMENT_REPRESENTATION_TYPE('digital', #80);
```

Example 41: exchange file segment for document type classification related to files

9.8.2 Document type classification for representations of documents

In the use for document master identification, the attribute product_data_type of document_type is constrained by the entity document_product_equivalence to take one of the following values: 'configuration_controlled_document', 'configuration_controlled_document_version', or 'configuration_ controlled_document_definition'. In consequence document_type cannot be used for the classification of document representations.

In AP214, the document type property information is instead captured in the description attribute of the related application_context. However, this mapping is the subject of several issues logged against this document. Refer to the Known Issues for this section (see 9.9.1).

The Instance Model: EXPRESS entities and attributes
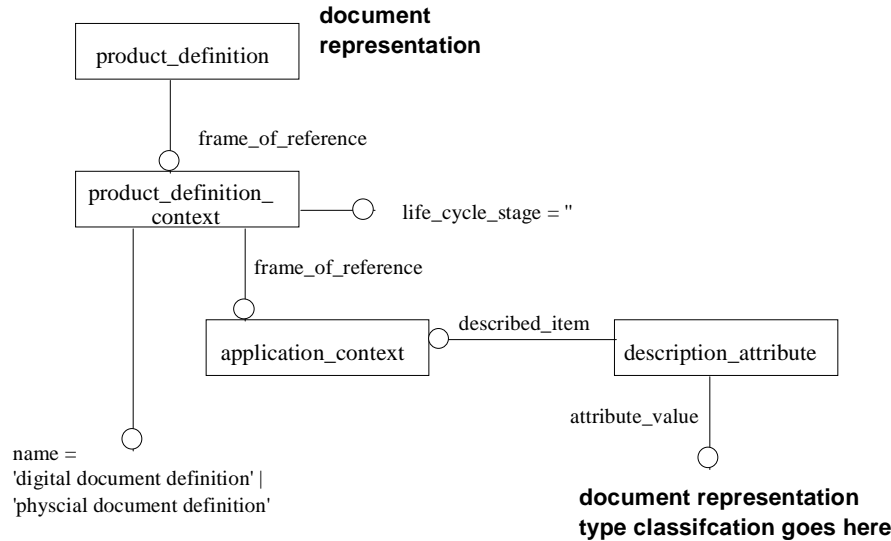
Diagram 41 : Instantiation diagram for document type classification of document representation

9.8.2.1 description_attribute

The entity description_attribute provides a capability to assign additional attribute/value pairs to be described entities. In the given context a description_attribute entity is used to associate document type classification information to the application context that is linked to the document representation (i.e. the product_definition) via its product_definition_context.

Attributes

- The *attribute_value* attribute provides the description text.
- The *described_item* attribute points to the application_context for which additional description is provided via the description_attribute.

ENTITY description_attribute	Attribute Population	Remarks
attribute_value	type: label = text = STRING	
described_item	type: entity = application_context	the application context related to the document representation

Preprocessor   Recommendations: Where   applicable   the   following   values   for description_attribute.attribute_value shall be used in the given context:
- '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 control planning data;
- 'drawing': The document represents a technical drawing.

Postprocessor Recommendations: None specific.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#50 = APPLICATION_CONTEXT('');
#70 = PRODUCT_DEFINITION_CONTEXT('digital document definition', #50,
'');
#75 = DESCRIPTION_ATTRIBUTE('geometry',#75);
#110 = PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS('dig1',
   'digital document 1', #10, #70, (#80));
```

Example 42: exchange file segment for document type classification for document representations

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('Xample file for document properties'), '2;1');
FILE_NAME('', '15.06.1999, 10:39:20', ('N.N.'), (''), '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
/* entities #10 to #70 represent a managed document with its context
**/
#10 = PRODUCT_DEFINITION_FORMATION('1', 'version 1', #20);
#20 = PRODUCT('doc1', 'cad-model',
   'managed document representing a CAD model', (#30));
#30 = PRODUCT_CONTEXT('', #50, '');
#40 = PRODUCT_RELATED_PRODUCT_CATEGORY('document', '', (#20));
#50 = APPLICATION_CONTEXT('');
#60 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema', 1998
   , #50);
#70 = PRODUCT_DEFINITION_CONTEXT('digital document definition', #50,
'');

/* Entities #75 to #110 represent the digital document with id 'file1'
*/
/* that is associated to the document #20 via #110
*/
#75 = DESCRIPTION_ATTRIBUTE('geometry',#230);
#80 = DOCUMENT_FILE('file1', '', $, #90, '', '');
#90 = DOCUMENT_TYPE('geometry');
#100 = DOCUMENT_REPRESENTATION_TYPE('digital', #80);
#110 = PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS('dig1',
   'digital document 1', #10, #70, (#80));

/* Entities #120 to #130 represent an external hardcopy that is
associated */
/* via the reference #125 to the part view defined by #550
*/
#120 = DOCUMENT_FILE('hardcopy', '', $, #90, '', '');
#125 = APPLIED_DOCUMENT_REFERENCE(#120,'',(#550));
#130 = DOCUMENT_REPRESENTATION_TYPE('physical', #120);

/* Entities #140 to #200 assign document format parameters to 'file1'
(#80) */
#140 = PROPERTY_DEFINITION('document property', '', #80);
#150 = PROPERTY_DEFINITION_REPRESENTATION(#140, #160);
#160 = REPRESENTATION('document format', (#170, #180, #190), #200);
```

```
#170 = DESCRIPTIVE_REPRESENTATION_ITEM('document format', 'VOXEL');
#180 = DESCRIPTIVE_REPRESENTATION_ITEM('size format', '5MB');
#190 = DESCRIPTIVE_REPRESENTATION_ITEM('character code', 'binary');
#200 = REPRESENTATION_CONTEXT('', 'document parameters');

/* Entities #210 to #250 assign document content parameters to 'file1'
(#80) */
#210 = PROPERTY_DEFINITION('document property', '', #80);
#220 = PROPERTY_DEFINITION_REPRESENTATION(#210, #230);
#230 = REPRESENTATION('document content', (#240, #250), #200);
#240 = DESCRIPTIVE_REPRESENTATION_ITEM('detail level', 'rough 3D
shape');
#250 = DESCRIPTIVE_REPRESENTATION_ITEM('geometry type', 'solid model');

/* Entities #210 to #250 assign document creation parameters to 'file1'
(#80)*/
#260 = PROPERTY_DEFINITION('document property', '', #80);
#270 = PROPERTY_DEFINITION_REPRESENTATION(#260, #280);
#280 = REPRESENTATION('document creation', (#290, #300, #310), #200);
#290 = DESCRIPTIVE_REPRESENTATION_ITEM('creating system', 'My CAD');
#300 = DESCRIPTIVE_REPRESENTATION_ITEM('operating system', 'Linux
2.1');
#310 = DESCRIPTIVE_REPRESENTATION_ITEM('creating interface',
   'export driver');

/* Entities #210 to #250 assign document format parameters to  */
/* the hardcopy document defined by #120                       */
#320 = PROPERTY_DEFINITION('document property', '', #120);
#330 = PROPERTY_DEFINITION_REPRESENTATION(#320, #340);
#340 = REPRESENTATION('document format', (#360), #200);
#360 = DESCRIPTIVE_REPRESENTATION_ITEM('size format', 'A0');

/* Entities #490 to #640 define the part master data and the
association */
/* of the documents to the part
*/
#490 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #500, #510);
#500 = PRODUCT_RELATED_PRODUCT_CATEGORY('Part', '', (#520));
#510 = PRODUCT_RELATED_PRODUCT_CATEGORY('Detail', $, (#520));

#520 = PRODUCT('mp1', 'my_part', '', (#30));
#530 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#520));
#540 = PRODUCT_DEFINITION_FORMATION('1', '', #520);
#550 = PRODUCT_DEFINITION('ddid', 'design view on my_part', #540,
#560);
#560 = PRODUCT_DEFINITION_CONTEXT('part definition', #50, '');
#590 = APPLIED_DOCUMENT_REFERENCE(#600, 'equivalence', (#550));
#600 = DOCUMENT('', '', '', #610);
#610 = DOCUMENT_TYPE('configuration controlled document version');
#620 = ROLE_ASSOCIATION(#630, #590);
#630 = OBJECT_ROLE('obligatory', '');
#640 = DOCUMENT_PRODUCT_EQUIVALENCE('equivalence', '', #600, #10);
ENDSEC;
END-ISO-10303-21;
```

Example 43 : Complete instantiation example for document properties

9.9 Known issues

9.9.1 Document type classification for managed documents

In AP214, the document type property information is instead captured in the description attribute of the related application_context. However, this mapping is the subject of several issues logged against this document. Attaching the document type property to the level of a document representation view definition is generally agreed not to be appropriate: the concept of a document type applies to all representation view definitions of all versions of the document.

10 Document and File Association with Product Data

In 'Document as Product' documents and external files may be associated with product data. This association is done in a consistent way using an applied reference with a specified role. The applied reference is realized structurally by the PDM Schema entities document and applied_document_reference.

When associating a managed 'Document as Product' to product data, the document master is linked to the applied reference constructs using the additional entity document_product_equivalence. This linkage may be made at the level of the base identification, the document version, or the document representation view definition. The recommended level from which a document master should reference other product data is the document version.

External files may also be associated with product data, in a way that is structurally consistent with that used for documents, using the entity applied_document_reference. Since external files are represented by the entity document_file (a subtype of document), only the applied_document_reference entity is required to associate an external file as a reference to other product data.

10.1 Document Reference

In 'Document as Product' documents may be associated with product data by reference in a specified role. The base document identification, the document version, or the document representation definition may serve as the point of assignment for a document master to be associated with other product data. It is generally recommended to make a document reference from the level of document version.

Reference of a managed document to product data is accomplished using the entities document_product_equivalence, document, document_type, and applied_document_reference.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements of document association with other product data are shown in Diagram 42.
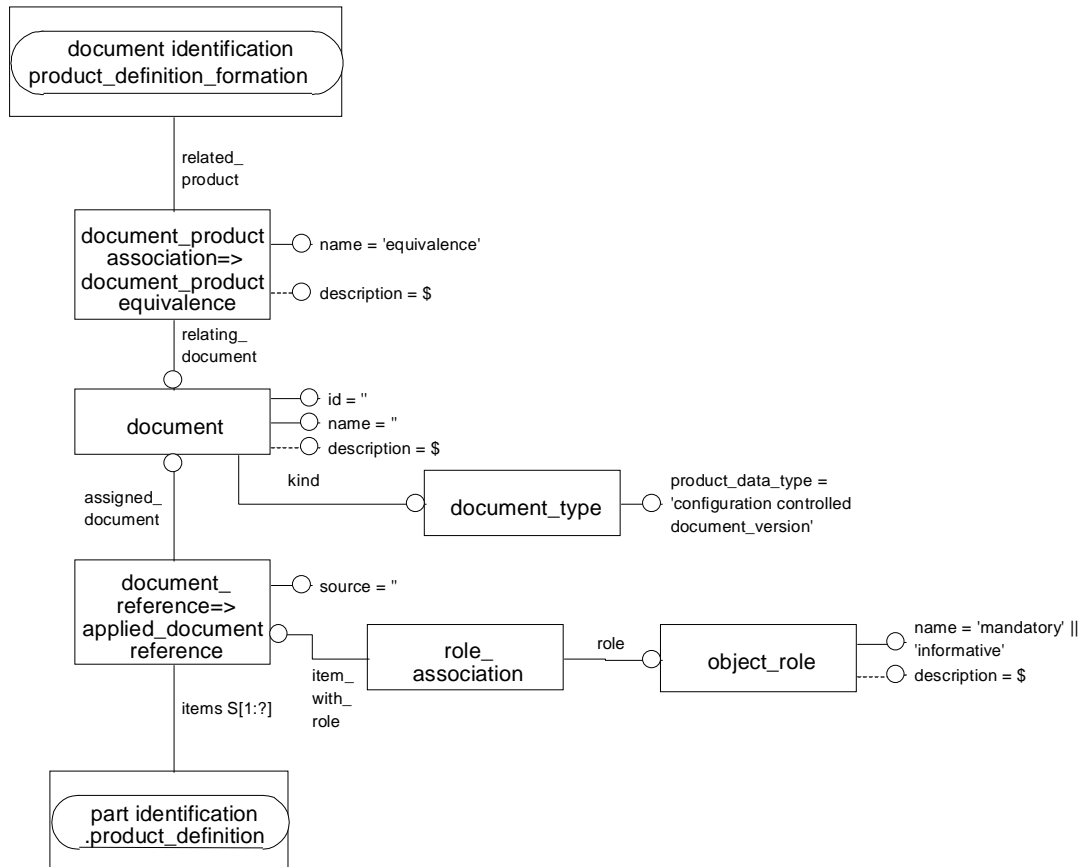
Diagram 42: Document Association to Product Data Instance Diagram

10.1.1.1   document_product_equivalence

This entity is a subtype of document_product_association relating a document to a 'product'. It asserts that the related product (or product_definition_formation or product_definition) in this case represents an element of a document master that is assigned as a reference to some other product data. The related_product attribute may refer to an instance of the entity product, product_definition_formation, or product_definition by way of the select type product_or_definition_or_formation. This provides the possibility to assign it to some other product data as a reference when used with the related entities document and applied_document_reference.

Attributes

- The *name* attribute characterizes the nature of this relationship.
- The *description* attribute is optional.
- The *related_product* attribute references the portion of the document master that is being assigned.
- The *relating_document* attribute has a document entity in a document_reference to some product data.

ENTITY document_product_equivalence	Attribute Population	Remarks
name	type: label = string 'equivalence'	
description	type: text = string	OPTIONAL
related_product	type: product_or_definition_or_formation	specifies the component of the document master that is the point

ENTITY document_product_equivalence	Attribute Population	Remarks
	(select)	of association to product data
relating_document	type: entity = document	

Preprocessor Recommendations: This entity should be used only when the requirement exists to associate a 'Document as Product' with product data. If the 'Document as Product' is not associated with product data, then this entity should not be instantiated.

The relating_document attribute always has a document entity as its value. Additional constraints apply to the document_type entity related to this document by the kind attribute. These prescribe a value for the product_data_type attribute of this document_type entity, correlated with the following select type values of the relating_document:

- Product   => 'configuration_controlled_document'
- Product_definition_formation => 'configuration_controlled_document_version'
- Product_definition   => 'configuration_controlled_document_definition'

Of these possible options for assignment, it is recommended to assign the 'configuration controlled document version' to product data. When assigned as a reference to a part master identification, it is recommended to assign the document version (product_definition_formation) to the view definition (product_definition) within the part master.

Postprocessor Recommendations: When importing an instance of this entity, postprocessors should recognize that the related_product attribute references a 'Document as Product'. The relating_document attribute references an instance of document that exists to allow assignment of the 'Document as Product' to product data.

Related Entities: There are no specific related entities.

10.1.1.2   document

This entity is only instantiated as itself when used as the relating_document in a document_product_equivalence relationship. In this role, the document entity is equated with the document master, and works together with the applied_document_reference to support the assignment of managed documents to product data in the STEP PDM Schema.

Attributes

- The ***kind*** attribute references a document_type entity.

ENTITY document	Attribute Population	Remarks
id	type: identifier = string	Not to be used for valid user data.
name	type: label = string	
description	type: text = string	OPTIONAL
kind	type: entity = document_type	The product_data_type attribute on this referenced entity is constrained by the entity document_product_equivalence

Preprocessor Recommendations: The document entity is only to be instantiated as itself in this role within the document_product_equivalence. In this case, the document identification is represented by the product master - the attributes on the document entity should not be used for identification of the managed document. It is not instantiated as itself, but as the subtype document_file when used to represent the identification of an external file reference (see 7.1).

Postprocessor Recommendations: The document entity is only a structural element in the assignment of a managed document to product data, along with the entity applied_document_reference. Identification of the managed document is represented by the product master entities according to the 'Document as Product' approach.

Related Entities: There are no specific related entities.

10.1.1.3   document_type

This entity is required for each instance of the entity document. In the given context, this entity is used to indicate that the related document objects are under configuration control. This corresponding usage of document_type is not to be used for a specific document object classification.

Attributes

- The *product_data_type* attribute describes the type of product data represented by the document entity.

ENTITY document_type	Attribute Population	Remarks
product_data_type	type: identifier = string	

Preprocessor Recommendations: In this use for document master identification, the attribute product_data_type is constrained by the entity document_product_equivalence to take one of the following values: 'configuration_controlled_document',   'configuration_controlled_document_version',   or 'configuration_controlled_document_definition'.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

10.1.1.4   applied_document_reference

This entity is a subtype of document_reference. It supports the assignment of documents to elements of product data within the select type document_reference_item. Notably, documents may be assigned to the part master view definition (product_definition) or to an individual occurrence of a part definition usage in an assembly structure (product_definition_relationship).

Attributes

- The *source* attribute is inherited from the supertype document_reference.
- The *items* attribute indicates the elements of product data referenced by the managed document.

ENTITY applied_document_reference	Attribute Population	Remarks
source	type: label = text	
assigned_document	type: entity = document	This referenced instance is the only case where the document entity is instantiated as itself, and not the subtype document_file.
items	type: document_reference_item = select	SET [1:?]

Preprocessor Recommendations: The inverse attribute role requires the associated entities role_association and object_role be instantiated related to this entity. Preprocessors should support as a minimum the assignment of document to the product_definition representing a part view definition.

Postprocessor Recommendations: Postprocessors should as a minimum recognize the assignment of document to the product_definition representing a part view definition.

Related Entities: There are no specific related entities.

10.1.1.5   role_association

This entity provides the item_with_role with a named role string. This is the method to add a role attribute to referenced entities that do not have one defined.

Attributes

- The *item_with_role* attribute.
- The *role* attribute references an instance of the entity object_role.

ENTITY role_association	Attribute Population	Remarks
item_with_role	type: role_select = select	
role	type: entity = object_role	

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

10.1.1.6   object_role

This entity assigns a role to the associated document reference. Two roles are defined:

- 'mandatory' means the assignment of the document to the product data is in a mandatory relationship - the document must be taken into account to understand the complete product information.
- 'informative' means the assignment of the document to the product data is an optional relationship - the document may be considered for additional information but is not required or enforced.

Attributes

- The *name* attribute indicates the name of the role.
- The *description* attribute is optional.

ENTITY object_role	Attribute Population	Remarks
name	type: label = string 'mandatory', 'informative'	
description	type: text = string	OPTIONAL

Preprocessor Recommendations: The name attribute should have the value 'mandatory' or 'informative' to indicate if the associated document reference is required or optional.

Postprocessor Recommendations: The value of the name attribute shall indicate if the document assignment is mandatory or optional for information only.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#70=PRODUCT('MP-03-2','my part',$,(#60));
#80=PRODUCT_DEFINITION_FORMATION('03','3rd modification',#70);
#90=PRODUCT_DEFINITION('/NULL',$,#80,#50);
...
#360=DOCUMENT_PRODUCT_EQUIVALENCE('equivalence',$,#370,#280);
#370=DOCUMENT('','',$,#380);
```

```
#380=DOCUMENT_TYPE('configuration controlled document version');
#390=APPLIED_DOCUMENT_REFERENCE(#370,'',(#90));
#400=ROLE_ASSOCIATION(#410,#390);
#410=OBJECT_ROLE('mandatory',$);
```

Example 44: exchange file segment for document association to product data

10.2 External File Reference

External files may also be associated with product data in a way that is consistent but simpler than that used for documents. While the requirement to assign documents to various product data is basically supported by three additional entities: document_product_equivalence, document, and applied_document_reference, external files need only the applied_document_reference entity to be related as a reference to other product data.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of external file association to product data are illustrated in Diagram 43.
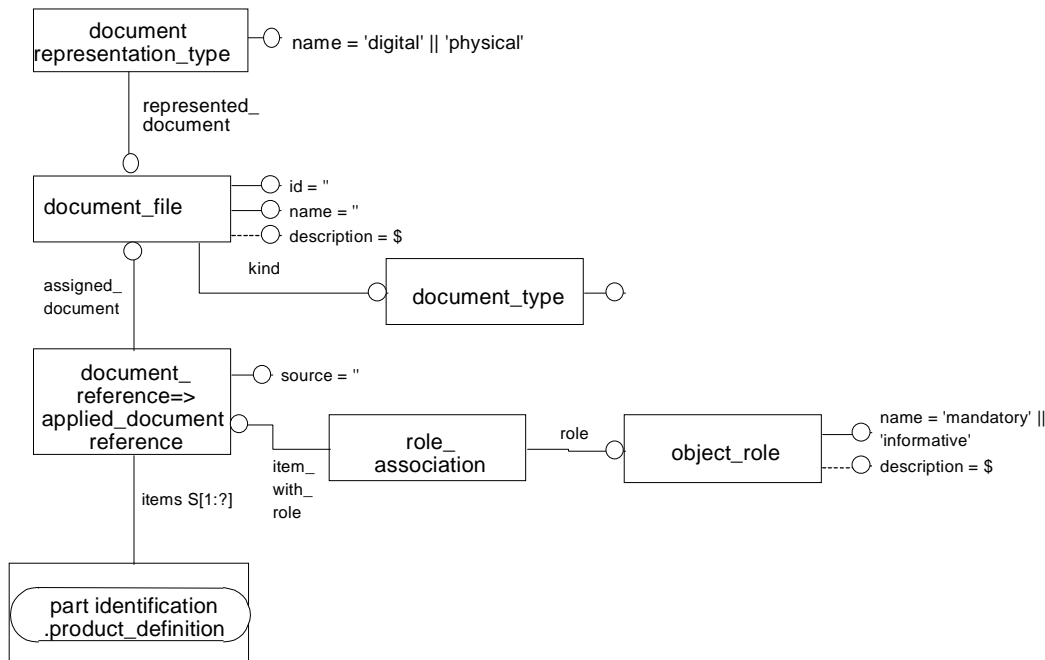
Diagram 43: External File Association to Product Data Instance Diagram

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#70=PRODUCT('MP-03-2','my part',$,(#60));
#80=PRODUCT_DEFINITION_FORMATION('03','3rd modification',#70);
#90=PRODUCT_DEFINITION('/NULL',$,#80,#50);
...
#120=DOCUMENT_TYPE('geometry');
#130=DOCUMENT_FILE('measure file id','measure data',$,#120,'','');
#140=APPLIED_DOCUMENT_REFERENCE(#130,'',(#90));
#150=OBJECT_ROLE('informative',$);
```

```
#160=ROLE_ASSOCIATION(#150,#140);
#170=DOCUMENT_REPRESENTATION_TYPE('digital',#130);
```

Example 45: exchange file segment for external file association to product data

10.3 Constrained Document or File Reference

Constraints may be specified on the association of documents or files with product data, in order to distinguish a portion of the entire document or file that applies in the reference.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of constrained document or file association to product data are illustrated in Diagram 44.
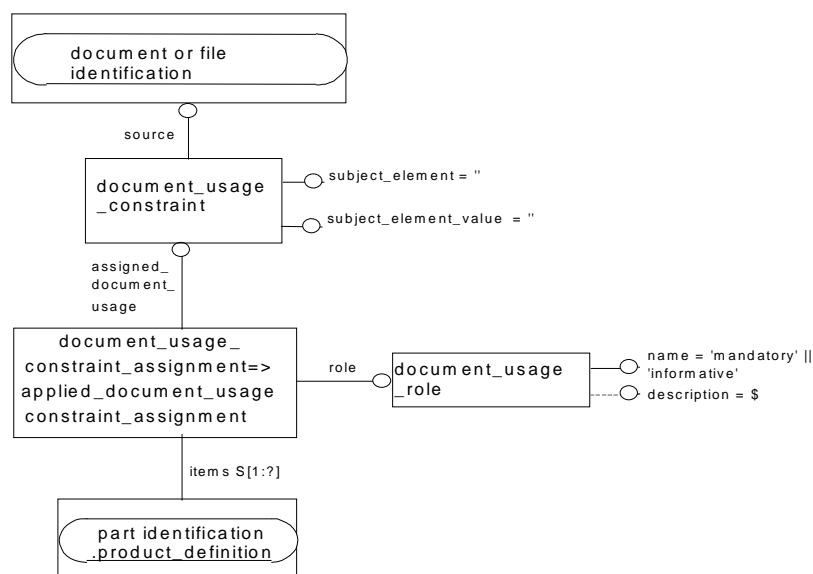
Diagram 44: Constrained Document Association to Product Data Instance Diagram

10.3.1.1   document_usage_constraint

This entity identifies a portion of a document that is applicable for a given usage.

Attributes

- The *source* attribute references the complete document or file of which a portion is distinguished.
- The *subject_element* attribute describes the portion or element of the complete document.
- The *subject_element_value* attribute conveys a specific value of the subject_element.

ENTITY document_usage_constraint	Attribute Population	Remarks
source	type: entity = document	May be document_file in the case of an external file reference
subject_element	type: label = string	
subject_element_value	type: text = string	

Preprocessor Recommendations: The subject_element should identify the relevant portion or section of the document that is applicable. The subject_element_value describes how the subject_element is to be interpreted.

Postprocessor Recommendations: Postprocessors should interpret the subject_element as an indication of the relevant portion of the entire document.

Related Entities: There are no specific related entities.

10.3.1.2   applied_document_usage_constraint_assignment

This entity is a subtype of document_usage_constraint_assignment. It supports the constrained assignment of a document to product data. A constrained assignment identifies only a portion of the entire document that is relevant in the assignment.

Attributes
- The *assigned_document_usage* attribute references the associated document_usage_constraint.
- The *role* attribute identifies the role of the assignment.
- The *items* attribute references the product data to which the partial document is assigned.

ENTITY applied_document_usage_constraint _assignment	Attribute Population	Remarks
assigned_document_usage	type: entity = document_usage_constraint	
role	type: entity = document_role	
items	type : docuement_reference_item = select	

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

10.3.1.3   document_usage_role

This entity identifies the role of the constrained document assignment.

Attributes
- The *name* attribute identifies the role of the document assignment.
- The *description* attribute is optional.

ENTITY document_usage_role	Attribute Population	Remarks
name	type : label = string 'mandatory', 'informative'	
description	type : text = string	OPTIONAL

Preprocessor Recommendations: The name attribute should contain the value 'mandatory' in case the document assignment is required, or 'informative' if the document assignment is optional.

Postprocessor Recommendations: Postprocessors should interpret the name attribute as an indication of whether the document assignment is required ('mandatory') or optional ('informative').

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#180=DOCUMENT_TYPE('geometry');
#190=DOCUMENT_FILE('plot_1','plot data',$,#180,'','');
#200=DOCUMENT_USAGE_CONSTRAINT(#190,'sheet','one');
#210=DOCUMENT_USAGE_ROLE('informative',$);
#220=APPLIED_DOCUMENT_USAGE_CONSTRAINT_ASSIGNMENT(#200,#210,(#90));
#230=DOCUMENT_REPRESENTATION_TYPE('physical',#190);
```

Example 46: exchange file segment for constrained document association to product data

The following section combines the above-discussed concepts and segments in a complete example. The concepts instantiated in the example are illustrated in Figure 11.
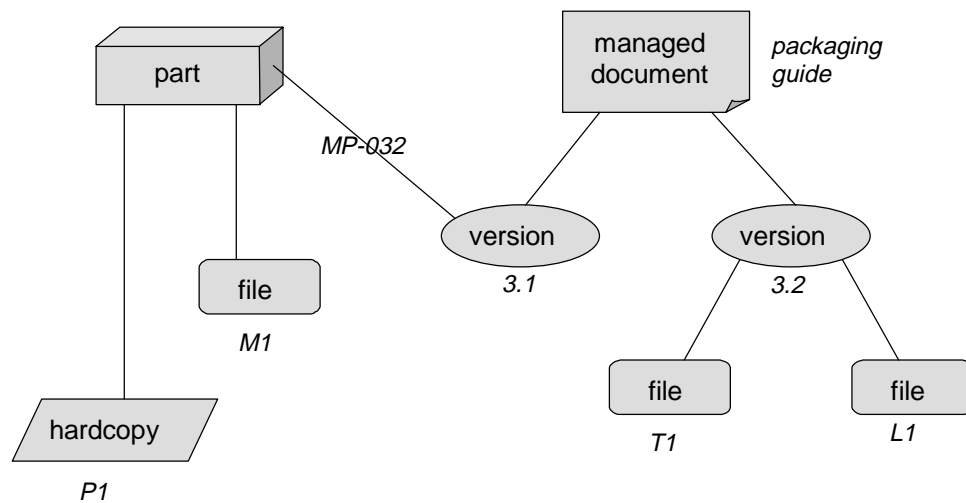
Figure 11: Schematic Overview of Complete Document and File Example

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of constrained document or file association to product data are illustrated in Diagram 45, Diagram 46, Diagram 47, and Diagram 48.
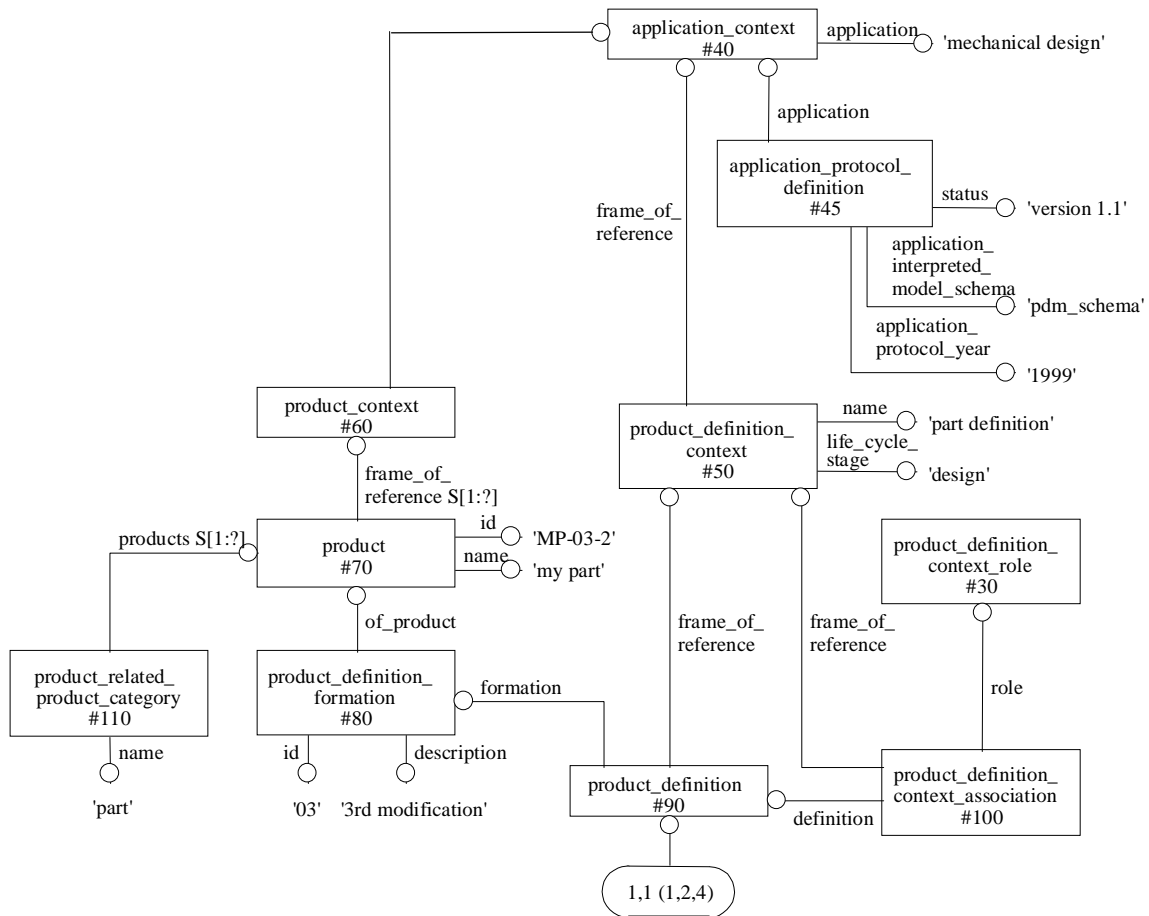
Diagram 45 : Part Master Instance Diagram
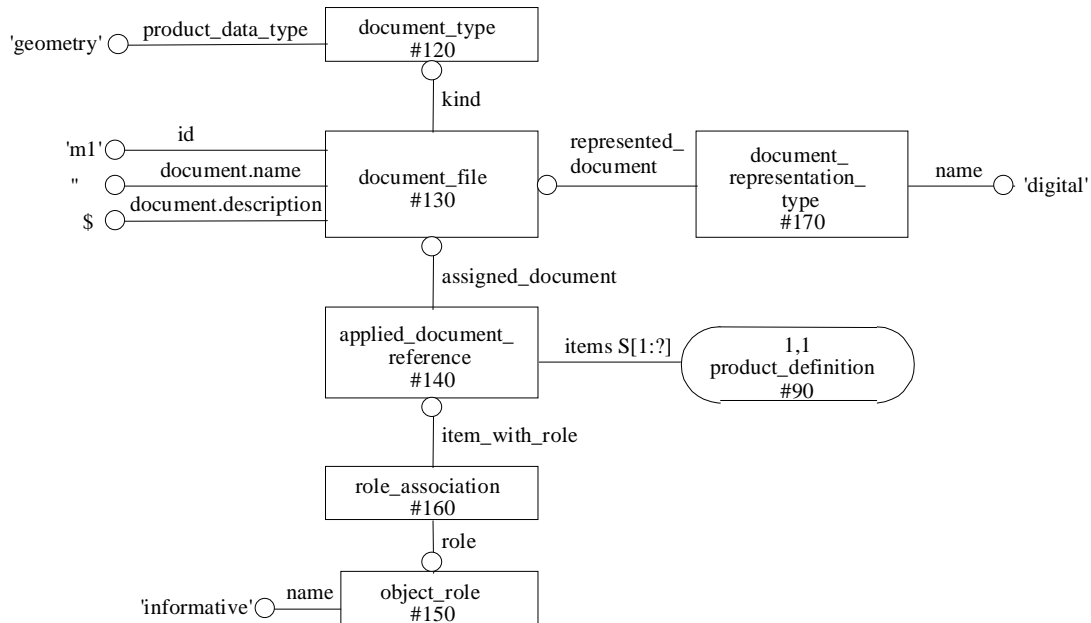
Diagram 46: External File Reference to Part View Definition Instance Diagram

'geometry' ○——product_data_type——— document_type #180

kind

'p1' ○———id———
" ○——document.name——— document_file #190 ———represented_document——— document_representation_type #230
$ ○——document.description——   'physical' ○—name—

source

'sheet' ○———subject_element———
'one' ○——subject_element_value—— document_usage_constraint #200   'informative' ○—name— document_usage_role #210

applied_document_usage

applied_document_usage_constraint_assignment #220   role

items S[1:?]

1,1
product_definition
#90

Diagram 47: Constrained External File Reference to Part View Definition Instance Diagram
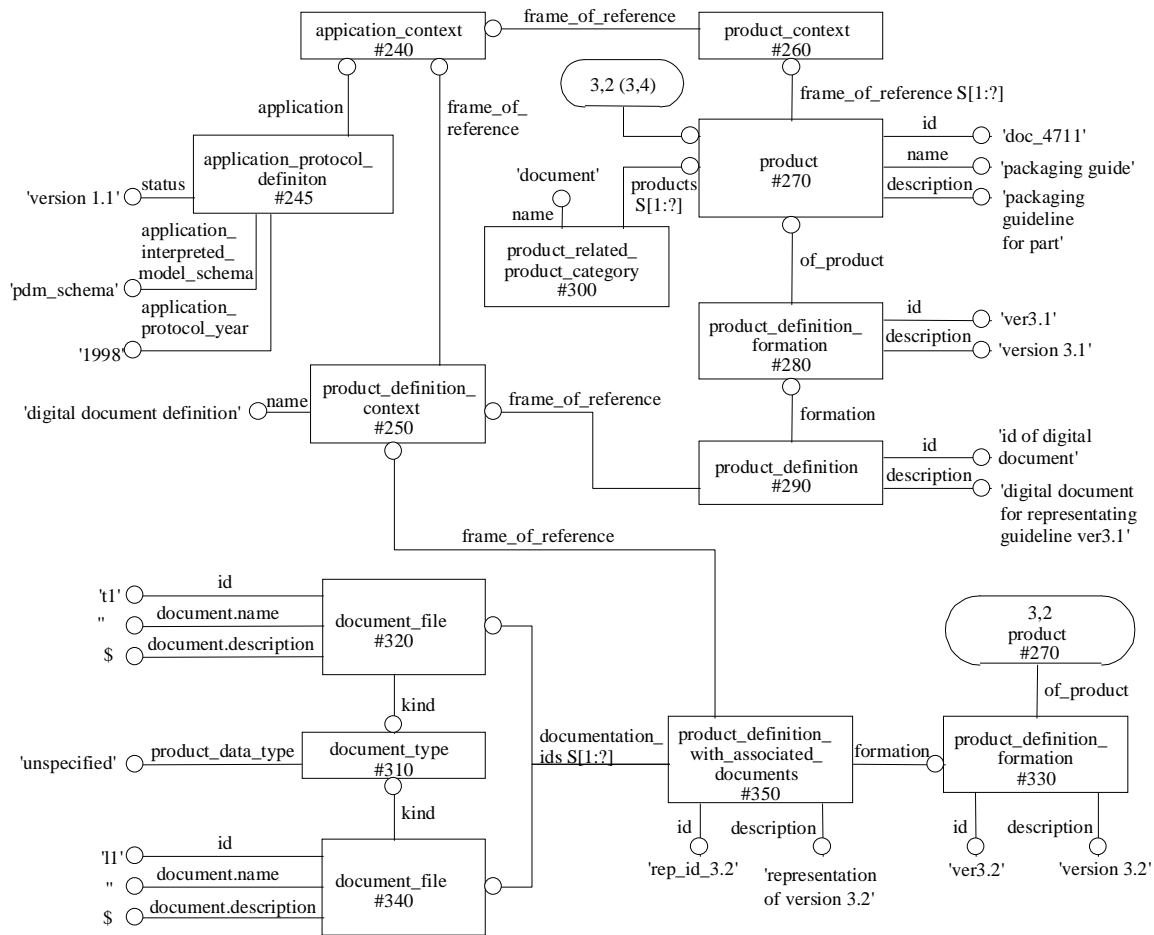
Diagram 48: Document and Constituent File Association Instance Diagram

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('pdm_schema data', 'pdm_schema'), '2;1');
FILE_NAME('C:document_xample.stp', '1999-5-20 T12:39:1', (''), (), '',
    '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;

DATA;
/*  Entities #10 - #20 define part master data                   */
/*  To the product_definition #90 is a life cycle specific view on  */
/*  that part to which documents and external files              */
/*  are assigned in the following sections of this file          */
#30=PRODUCT_DEFINITION_CONTEXT_ROLE('',$);
#40=APPLICATION_CONTEXT('mechanical design');
#45=APPLICATION_PROTOCOL_DEFINITION('version
1.1','pdm_schema',1998,#40);
#50=PRODUCT_DEFINITION_CONTEXT('part definition',#40,'design');
#60=PRODUCT_CONTEXT('',#40,'');
```

```
#70=PRODUCT('MP-03-2','my part',$,(#60));
#80=PRODUCT_DEFINITION_FORMATION('03','3rd modification',#70);
#90=PRODUCT_DEFINITION('/NULL',$,#80,#50);
#100=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#90,#50,#30);
#110=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#70));

/*  Entities #120 to #170 model the assignment of                */
/* digital file data to the product_definition #90               */
#120=DOCUMENT_TYPE('geometry');
#130=DOCUMENT_FILE('m1','',$,#120,'','');
#140=APPLIED_DOCUMENT_REFERENCE(#130,'',(#90));
#150=OBJECT_ROLE('informative',$);
#160=ROLE_ASSOCIATION(#150,#140);
#170=DOCUMENT_REPRESENTATION_TYPE('digital',#130);

/* Entities #180 to #230 model the asignment of hardcopy document    */
/* data to the product_definition #90                                */
/*  This assignment is partial, i.e., in addition the information is */
/*  conveyed that only sheet one of the hardcopy (plots) shall be    */
/*  logically assigned                                               */
#180=DOCUMENT_TYPE('geometry');
#190=DOCUMENT_FILE('p1','',$,#180,'','');
#200=DOCUMENT_USAGE_CONSTRAINT(#190,'sheet','one');
#210=DOCUMENT_USAGE_ROLE('informative',$);
#220=APPLIED_DOCUMENT_USAGE_CONSTRAINT_ASSIGNMENT(#200,#210,(#90));
#230=DOCUMENT_REPRESENTATION_TYPE('physical',#190);

/* Entities #240 to #350 model a managed document (document      */
/* as product)  For this document two versions are specified     */
/*(#280 and #330)To version 3.2 (#330) two files (#320, #340)    */
/* that constitute the document are defined                      */
#240=APPLICATION_CONTEXT('');
#250=PRODUCT_DEFINITION_CONTEXT('digital document definition',#240,'');
#260=PRODUCT_CONTEXT('',#240,'');
#270=PRODUCT('doc_4711','packaging guide','packaging guideline for
part',(#260));
#280=PRODUCT_DEFINITION_FORMATION('ver3.1','version 3.1',#270);
#290=PRODUCT_DEFINITION('id of digital document','digital document for
representing guideline ver3.1',#280,#250);
#300=PRODUCT_RELATED_PRODUCT_CATEGORY('document',$,(#270));
#310=DOCUMENT_TYPE('unspecified');
#320=DOCUMENT_FILE('t1','',$,#310,' ',' ');
#330=PRODUCT_DEFINITION_FORMATION('ver3.2','version 3.2',#270);
#340=DOCUMENT_FILE('l1','','file with logo for the guide',#310,'','');
#350=PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS('rep_id_3.2','represe
ntation of version 3.2',#330,#250,(#340,#320));

/* Entities #360 to #410 establish the association of the        */
/* managed document version #280 (packaging guide) with the view */
/* of the part modeled via product_definition #90                */
#360=DOCUMENT_PRODUCT_EQUIVALENCE('equivalence',$,#370,#280);
#370=DOCUMENT('','',$,#380);
#380=DOCUMENT_TYPE('configuration controlled document version');
#390=APPLIED_DOCUMENT_REFERENCE(#370,'',(#90));
#400=ROLE_ASSOCIATION(#410,#390);
#410=OBJECT_ROLE('mandatory',$);
```

```
ENDSEC;
END-ISO-10303-21;
```

Example 47: exchange file for complete document and file identification and association

11 Document and File Relationships

This section discusses relationships between files, document representation and sequence structure relationships between versions of managed documents.

The PDM Schema supports the definition of document structure by defining relationships of the representations of the documents using the entity product_definition_relationship. The PDM Schema supports the definition of corresponding relationships between document_file instances via the entity document_relationship. In addition, the PDM Schema supports the definition of relations between versions of a managed document to model document history.

11.1 'Sequence' relationships between document versions

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of part version sequence history are illustrated in Diagram 31.
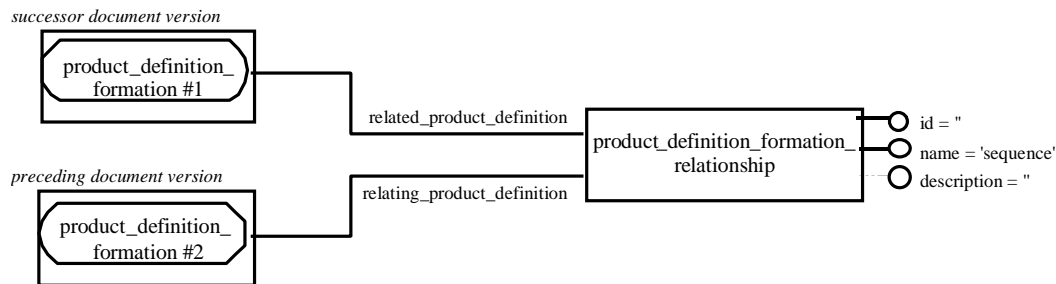
Diagram 49 : Document Version (sequence) History Instance Diagram

11.1.1.1   product_definition_formation_relationship

In this context the entity product_definition_formation_relationship is used to establish sequence relationships between product_definition_formations instances that represent document versions. In consequent interpretation of the 'document as product' approach the entity structure is exactly the same as used in the approach for part version history.

Attributes
- The *id* attribute provides an identifier of the relationship.
- The *name* attribute specifies this relationship represents a version 'sequence' history.
- The *relating_product_definition_formation* attribute references the preceding part version .
- The *related_product_definition_formation* attribute references the subsequent part version.

ENTITY product_definition_formation_ relationship	Attribute Population	Remarks
id	type: identifier = string	
name	type: label = string	shall be 'sequence'
description	type: text = string	OPTIONAL
related_product_definition_formation	type : entity = product_definition_formation	subsequent document version
relating_product_definition_formation	type : entity = product_definition_formation	preceding document version

Preprocessor Recommendations: The value 'sequence' for the name attribute identifies this as the version sequence history relationship. There is no standard mapping for the description attribute in a product_definition_formation_relationship. The id attribute must be unique with respect to the relationship, but there is no standard mapping for the value.

Postprocessor Recommendations: Since there are no standard mapping for the description attributes for a product_definition_formation_relationship, it is recommended that postprocessors not assign any processing significance to this value.

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context for design assembly life cycle */
#1000 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #1000, '');
#230 = PRODUCT_DEFINITION_CONTEXT('document definition', #1000,
'design');

/* product type discrimination */
/* #100 identifies the instance #2 as representing a document */
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('document',$,(#2));

/* version A - preceding version */
#2=PRODUCT('11000','a document','description for doc 11000',(#220));
#3=PRODUCT_DEFINITION_FORMATION('A', 'version A for the document',#2);
#4=PRODUCT_DEFINITION('D1','detailed drawing',#3,#230);

/* version B - subsequent version */
#10=PRODUCT_DEFINITION_FORMATION('B', 'version B for the document
11000',#2);
#11=PRODUCT_DEFINITION('D2','detailed drawing',#10,#230);

#16=PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('id','sequence',$,#3,#10)
;
```

Example 48 : exchange file for document version history

11.2 Relationships between document representations

Product_definition_relationship associates in the given context two instances of product_definition that model a representation of a document. The semantics of the relationship are defined by the attributes name and description of the entity product_definition_relationship. The name attribute of product_definition_relationship defines the relation type.

The Instance Model: EXPRESS entities and attributes

The figure below shows the general structure to link two product_definitions that model document representations. The semantics of the relation is defined by the string provided in product_definition_relationship.name.
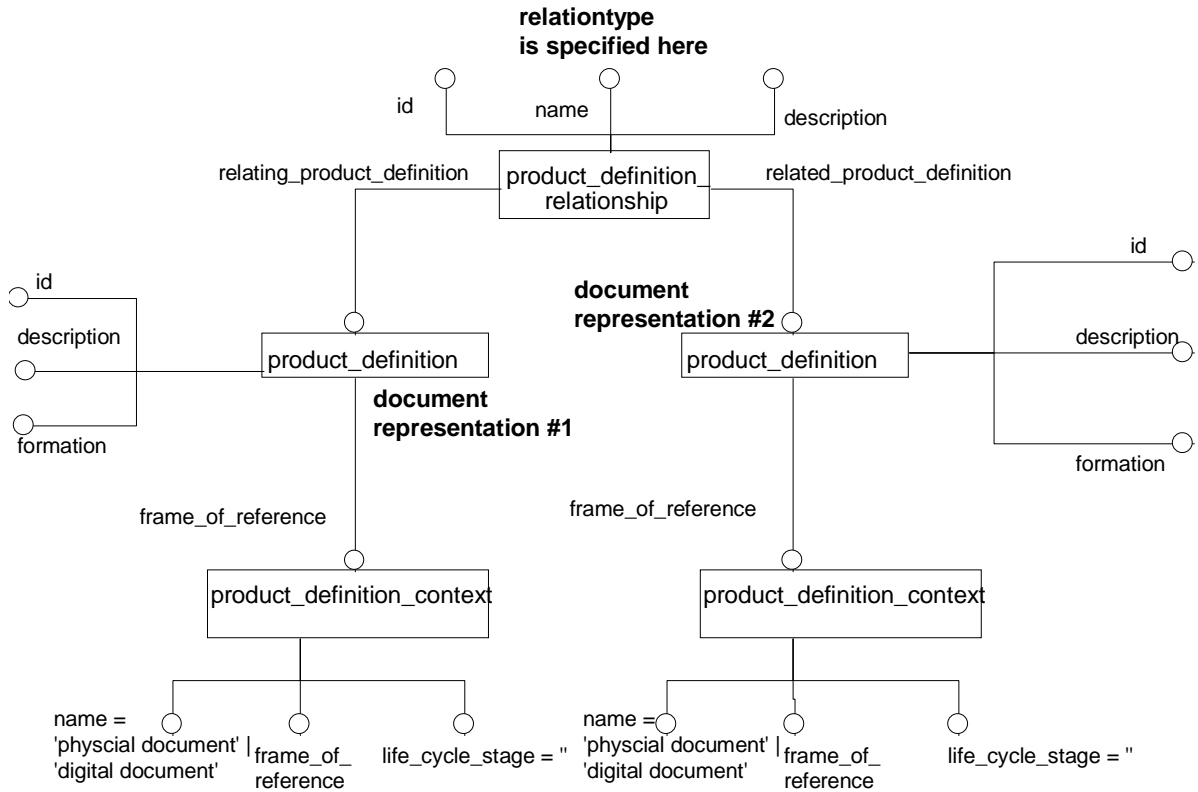
Diagram 50: Document representation relationship Instance Diagram

11.2.1.1   product_definition_relationship

Attributes
- The *id* is an identifier that distinguishes the product_definition_relationship.
- The *name* is the label by which the product_definition_relationship is known.
- The *description* is an optional text that characterizes the product_definition_relationship.
- The *relating_product_definition* contains one of the product_definition entity types, which is a part of the relationship.
- The *related_product_definition* is the other product_definition, which is part of the relationship. If one attribute is dependent upon the other, this attribute shall be the dependent one.

ENTITY product_definition_relationship	Attribute Population	Remarks
id	type: identifier = string	should be instantiated as ''
name	type: label = string	defines the semantics of the relation (see table below)
description	type: text = string	OPTIONAL
relating_product_definition	entity: product_definition	
related_product_definition	entity: product_definition	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: the usage of product_definition_relationship described here relates two product_definitions that define document representations.

Where applicable the following values shall be used for product_definition_relationship.name:

- 'addition': specifies that the related document provides supplementary or collateral information with regard to the information provided by the relating document;
- 'copy': defines a relationship where the related document representation is a copy of the relating document representation;
- 'derivation': defines a relationship where the related document representation is derived from the relating document representation (e.g. a raster image may be derived from a 3D CAD model);
- 'decomposition': defines a relationship where the related document representation is one of potentially more sub documents of the relating document representation (e.g. a decomposition of a document into clauses or into several CAD models);
- 'peer': specifies that the related document provides required information with regard to that provided by the relating document. The peer document is essential for a complete understanding;
- 'reference': defines a relationship where the related document is referenced from the relating, e.g., a text processor representation referencing a CGM representation for inclusion as a graphical object in the text document;
- 'sequence': defines a logical sequence where the related document representation comes after the relating document representation (e.g. a sequence of clauses);
- 'substitution': defines a relationship where the related document representation replaces the relating document representation;
- 'translation': specifies that the related document is generated automatically from the relating document.

The relation types 'decomposition' and 'substitution' can be used to model 'assembly'-type structures of document representations. This approach does not mirror the approach taken with assembly_component_usage for the product structure. This reflects the understanding that there is no common requirement to manage document representation structures in the same way as product structure.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('example for document rep structures'),
    '2;1');
FILE_NAME('', '20.08.1999, 10:16:08', ('n.n.'), (''),
    '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
/* Entities #10 to #60 define the meta data for a managed document */
#10 = PRODUCT_DEFINITION_FORMATION('1', '', #20);
#20 = PRODUCT('d1', 'doc1', 'document nr 1', (#40));
#30 = PRODUCT_RELATED_PRODUCT_CATEGORY('document', '', (#20));
#40 = PRODUCT_CONTEXT('', #50, '');
#50 = APPLICATION_CONTEXT('');
#60 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema', 1998
    , #50);

/* Entities #70 to #90 defines two digital files. The digital file #70
*/
/* represents a text procoessor file with included graphics. The
digital */
```

```
/* file #90 represents an image file to be included in #70
*/
#70 = PRODUCT_DEFINITION('doc1_master', 'text file including graphics
for doc1', #10, #80);
#80 = PRODUCT_DEFINITION_CONTEXT('digital document definition', #50,
'');
#90 = PRODUCT_DEFINITION('doc1_image', 'image to be included in doc1',
#10, #80);

/* Entity #100 defines that the file #90 (image) is referenced by the
*/
/* file #70 (text master file)
*/
#100 = PRODUCT_DEFINITION_RELATIONSHIP('','reference',$,#70,#90);

/* other product data follows */

ENDSEC;
END-ISO-10303-21;
```

Example 49: Document representation relationship instantiation

11.3 Relationships between external files

Document_relationship associates in the given context two instances of document files. The semantics of the relationship are defined by the attributes name and description of the entity document_relationship. The name attribute of document_relationship defines the relation type. The semantics of the relation types mirror the capability as defined for the relationships between document representations.

The Instance Model: EXPRESS entities and attributes

The generic structure to establish a relationship between document_files is illustrated in Diagram 51. The semantics of such a relation is identified via document_relationship.name.
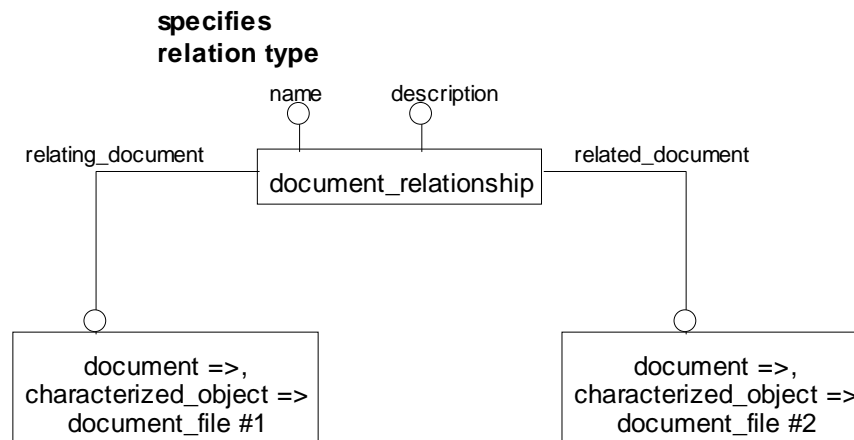
Diagram 51: relation between document files

11.3.1.1   document_relationship

Attributes

- The *name* defines the semantics of the document_relationship.
- The *description* is an optional text that characterizes the document_relationship.
- The *relating_document* contains one of the document file entity types, which is a part of the relationship.
- The *related_document* is the other document file, which is part of the relationship. If one attribute is dependent upon the other, this attribute shall be the dependent one.

ENTITY document_relationship	Attribute Population	Remarks
name	type: label = string	defines the semantics of the relation (see table below)
description	type: text = string	OPTIONAL
relating_document	entity: document	in this context the subtype document_file of document is used
related_document	entity: document	in this context the subtype document_file of document is used

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: A document_relationship entity relates two document_file instances.

Where applicable the following values shall be used:

- 'addition': specifies that the related document provides supplementary or collateral information with regard to the information provided by the relating document;
- 'copy': defines a relationship where the related document file is a copy of the relating document file;
- 'derivation': defines a relationship where the related document file is derived from the relating document file (e.g. a raster image may be derived from a 3D CAD model);
- 'decomposition': defines a relationship where the related document file is one of potentially more sub documents of the relating document file (e.g. a decomposition of a document into clauses or into several CAD models);
- 'peer': specifies that the related document provides required information with regard to that provided by the relating document. The peer document is essential for a complete understanding;
- 'reference': defines a relationship where the related document is referenced from the relating, e.g., a text processor file referencing a CGM file for inclusion as a graphical object in the text document;
- 'sequence': defines a logical sequence where the related document file comes after the relating document file (e.g. a sequence of clauses);
- 'substitution': defines a relationship where the related document file replaces the relating document file;
- 'translation': specifies that the related document is generated automatically from the relating document.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''),'2;1');
FILE_NAME('', '20.08.1999, 13:19:42', (''), (''),
   '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
```

```
DATA;
#10 = PRODUCT_DEFINITION_FORMATION('1', '', #20);
#20 = PRODUCT('doc', 'versioned_doc', 'a managed document', (#40));
#30 = PRODUCT_RELATED_PRODUCT_CATEGORY('document', '', (#20));
#40 = PRODUCT_CONTEXT('', #50, '');
#50 = APPLICATION_CONTEXT('');
#60 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema', 1998
, #50);
#80 = PRODUCT_DEFINITION_CONTEXT('digital document definition', #50,
'');
#90 = DOCUMENT_FILE('file_1', '', '', #110, ' ', ' ');
#100 = DOCUMENT_REPRESENTATION_TYPE('digital', #90);
#110 = DOCUMENT_TYPE('');
#120 = PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS('dig_doc',
   'digital document for doc', #10, #80, (#90, #130));
#130 = DOCUMENT_FILE('file_2', '', '', #110, ' ', ' ');
#140 = DOCUMENT_REPRESENTATION_TYPE('digital', #130);

#150 = DOCUMENT_RELATIONSHIP('sequence','defines a sequence relation',
      #90,#130);

/* other product data follows */

ENDSEC;
END-ISO-10303-21;
```

Example 50 : exchange file for document file relationship

12 Alias Identification

An alias identification is a mechanism to associate an object with an additional identifier that is used to identify the object of interest in a different context, either in another organization, or in some other context. The alias identification mechanism shall not be used to alias supplied parts.

The scope of the alias identification shall be specified either by the description of the associated identification_role or – if the scope is defined by an organization – with help of an applied_organization_assignment. The scope of an alias defines the context in which the id specified via applied_identification_assignment.assigned_id overrides the original id. A scenario might be that an object has an id in the context of the organization assigned in the role 'id owner' as a primary id and other ids defined via aliases that are valid in the context of some other organizations.

The Instance Model: EXPRESS entities and attributes

The structure the the alias assignment is illustrated in Diagram 52. The applied_identification_assignment #1 implements the alias via pointing to the to be aliased objects via applied_identification_assignment.items. The alias is represented in the attribute apllied_identification_assignment.assigned_id. To mark the applied_identification_assignment as an alias an instance of identification_role with name = 'alias' is attached to it.
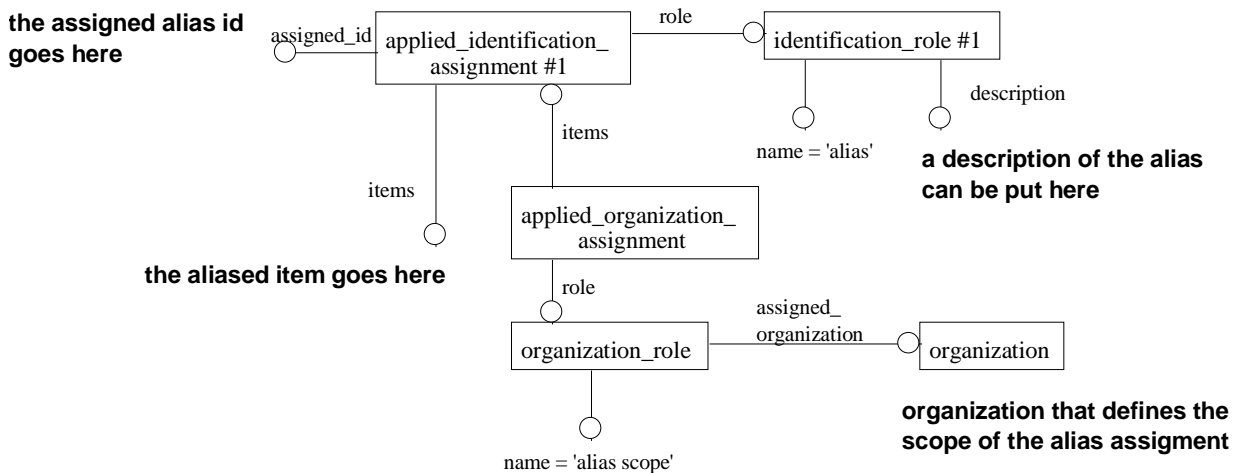
Diagram 52: Alias Identification Instance Diagram

The following table list the objects to which aliases can be assigned in the PDM Schema:

entity type	purpose of usage as aliased item (applied_identification_assignment.items)
approval_status	allows to alias the name for the status of acceptance, i.e. approval_status.name
document_file	allows to alias the id of a digital or physical file, i.e. document_file.id
organization	allows to provide an alias for organization.id
product_definition	allows to provide an alias identification for product_definition.id. this capability can be used for instances of product_definition that model document representations as well as for production_definitions that represent views on parts

12.1.1.1   applied_identification_assignment

This entity is a subtype of the entity identification_assignment. It allows the actual assignment of an identification_assignment entity to product data, in this case to a document_file entity representing an external file.

Attributes
- the *assigned_id* provides the identification string that is assigned to the objects listed in the items attribute.
- the *role* references an instance of identification_role that defines the role of the assignment. In the context here this role has the name 'alias'.
- The *items* attribute is used to reference the associated external file.

ENTITY applied_identification_ assignment	Attribute Population	Remarks
assigned_id	type: identifier = string	
role	type: entity = identification_role	
items	type : entity =	

Preprocessor Recommendations: For each applied_identification_assignment used as an alias identification an identification_role with name 'alias' must be used.

Postprocessor Recommendations: None specific.

Related Entities: If the scope of an alias is an organization, this scope can be specified by using an applied_organization_assignment that points to the applied_identification_assignment, which represents the alias.

12.1.1.2   identification_role

This entity defines a role for an identification_assignment and a description of that role. In the scope of the alias concept the identification_role has to be labeled 'alias'.

Attributes
- The *name* is the label by which the identification_role is known.
- The *description* is an optional text that characterizes the identification_role.

ENTITY identification_role	Attribute Population	Remarks
name	type: label = string 'version'	must be instantiated as 'alias'
description	type: text = string	OPTIONAL

Preprocessor Recommendations: The description attribute can be used to specify the applicability of an alias – if not specified by an organisational scope. An example for such would be a alias that is used internally in one specific system. '

Postprocessor Recommendations: None specific.

*Related Entities***:** The entity identification_role describes the role of an applied_identification_assignment. In the context of alias definition this role has to be name 'alias'.

12.1.1.3   applied_organization_assignment

This entity is a subtype of organization_assignment, allowing the representation of the assignment of an organization to some product data.

Attributes
- The *assigned_organization* is the organisation which is to be associated with the product data.
- The *role* attribute specifies the purpose of the association of the organization_assignment with product data. In the context of using this entity for alias identification the role is 'alias scope.
- The *items* attribute is a reference to the product data to which the organization is being assigned.

ENTITY applied_organization_assignment	Attribute Population	Remarks
assigned_organization	type: entity = organization	
role	type: entity = organization_role	must be 'alias scope' in the given scope
items	type: entity (defined by organization_item_select)	SET[1:?]

Preprocessor Recommendations: For defining an organizational scope of an alias assignment the applied_organization_assignment shall have a role 'alias scope'.

Postprocessor Recommendations: None specific.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''),'2;1');
FILE_NAME('', '20.08.1999, 13:19:42', (''), (''),
   '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
/* The entities in the following section describe product data and  */
/* organisational data as described in other chapters of the PDM    */
/* usage guide                                                      */
#10 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #20, #30);
#20 = PRODUCT_RELATED_PRODUCT_CATEGORY('Part', '', (#40));
#30 = PRODUCT_RELATED_PRODUCT_CATEGORY('Detail', $, (#40));
#40 = PRODUCT('p1', 'part 1', '', (#60));
#50 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#40));
#60 = PRODUCT_CONTEXT('', #70, '');
#70 = APPLICATION_CONTEXT('');
#80 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema', 1998
   , #70);
#90 = PRODUCT_DEFINITION_FORMATION('1', '', #40);
#100 = PRODUCT_DEFINITION('v1', 'view_on_part_1', #90, #110);
#110 = PRODUCT_DEFINITION_CONTEXT('part definition', #70, '');
#120 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#100, $, #130);
#130 = PRODUCT_DEFINITION_CONTEXT_ROLE('', $);
#140 = ORGANIZATION('org1', 'My organization', 'company');
#190 = ORGANIZATIONAL_ADDRESS('', '', '', '', 'ducktown', '', '', '',
   ''
```

```
   , '', '', '', (#140), 'postal address');
#200 = APPLIED_ORGANIZATION_ASSIGNMENT(#140, #210, (#40));
#210 = ORGANIZATION_ROLE('id owner');

/*entities #150 to #185  express that the id of organization #140 */
/* is 'xyz_org' inside organization #185                         */
#150 = APPLIED_IDENTIFICATION_ASSIGNMENT('xyz_org', #180, (#140));
#160 = APPLIED_ORGANIZATION_ASSIGNMENT(#185, #170, (#150));
#170 = ORGANIZATION_ROLE('alias scope');
#180 = IDENTIFICATION_ROLE('alias', 'alias for org1');
#185 = ORGANIZATION('org2', 'My second organization', 'company');

/* Entities #220 to #240 define that the id 'v1' of the part view #100
*/
/* is aliased with the id 'v_external_1' inside the organization #140
*/
#220 = APPLIED_IDENTIFICATION_ASSIGNMENT('v_external_1', #240, (#100));
#230 = APPLIED_ORGANIZATION_ASSIGNMENT(#140, #235, (#220));
#235 = ORGANIZATION_ROLE('alias scope');
#240 = IDENTIFICATION_ROLE('alias', 'alias for view id');

/* Entities #250 to ##270 define that the information related to the */
/* part view #100 has the status 'approved'                          */
#250 = APPROVAL(#270, '');
#260 = APPLIED_APPROVAL_ASSIGNMENT(#250, (#100));
#270 = APPROVAL_STATUS('approved');

/* Entities #280, #290 define 'accepted' as an alias for the approval
*/
/* status 'approved'                    */
#280 = APPLIED_IDENTIFICATION_ASSIGNMENT('accepted', #290, (#270));
#290 = IDENTIFICATION_ROLE('alias', 'alias for approval status -
approved -');

ENDSEC;
END-ISO-10303-21;
```

Example 51 : exchange file for alias identification

12.2 Known issues

12.2.1 Restricted scope for alias identification in the PDM Schema

In contrast to AP214, the PDM Schema does not allow alias identifications of properties, parts, (external) geometric models, managed documents. An extension of the PDM Schema to support additional capability in this area will be investigated.

13 Authorization

13.1 Organization and Person

The PDM Schema represents organizations and people in organizations as they perform functions related to other product data and data relationships. A person in the PDM Schema must exist in the context of some organization. An organization or a person in an organization is then associated with the data or data relationship in some role indicating the function being performed. Both people and organizations may have addresses associated with them. The address is entirely optional; it is done through the address entity being related to the person (through personal_address) or organization (through organizational_address).

13.1.1 Organization

The PDM Schema represents groups of people (e.g., companies, countries, etc.) through the organization entity. The identification or id data is optional. However, because this information is very important in providing unique identification to the organization or company, it is recommended that this field always be populated with unique data. If appropriate, a URL-like convention for the organization identifier may be useful, e.g., prostep.de. The name attribute should contain the common nomenclature of the organization. The description attribute may contain a characterization of the type of the organization, or a textual explanation of its reason for existence.

The organization entity is related to certain constructs to identify the organizations responsible for them and how they are responsible. This relationship is defined through the applied_organization_assignment entity (which relates an organization in some role) to an entity. The role is established in the organization_role entity name attribute. The sections, which describe the use of the entity to which the organization is assigned, will identify the allowed values for the name attribute of the organization_role entity. The organization_role entity may have a description associated with it through the entity description_attribute and its attribute_value attribute.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements for organization are illustrated in Diagram 53.

Diagram 53: Organization Instance Diagram

13.1.1.1   organization

Entity represents the identification of an identified group of people gathered and structured for a purpose.

Attributes
- The *id* attribute is the identifier that distinguishes the organization.
- The *name* attribute is the label by which the organization is known.
- The *description* attribute characterizes the type of organization, or an explanation of its reason for existence.

ENTITY organization	Attribute Population	Remarks
id	type: identifier = string	OPTIONAL it is recommended to instantiate this attribute
name	type: label = string	
description	type: text = string	OPTIONAL if available the type of organization shall be mapped into this attribute

Preprocessor Recommendations: All preprocessors should provide a unique organization id to eliminate ambiguities where organizations may have the same names. If the intended domain for the data is large, the reader is referred to ISO/IEC 8824-1, which can provide some guidance on creating unique identifiers. If appropriate, a URL like convention for the organization identifier may be used, e.g., prostep.de. A unique string obtained under ISO/IEC 8824-1 can be used as, or prefixed to, the organization identifier. For example, if the organization typically used an identifier of "93699" and the unique string were "USA", the unique value of the organization id would be "USA93699". If available and appropriate, the following values should be used to describe the organization type: 'company' to indicate a business entity; 'department' to indicate an orgaqnizational group within a company.

Postprocessor Recommendations: All postprocessors should make use of any provided information in the id attribute to eliminate ambiguities where organizations may have the same name.

Related Entities: There are no specific related entities.

13.1.1.2   applied_organization_assignment

This entity is a subtype of organization_assignment, allowing the representation of the assignment of an organization to some product data.

Attributes
- The *assigned_organization* is the organization which is to be associated with the product data.
- The *role* attribute specifies the purpose of the association of the organization_assignment with product data.
- The *items* attribute is a reference to the product data to which the organization is being assigned.

ENTITY applied_organization_assignment	Attribute Population	Remarks
assigned_organization	type: entity = organization	
role	type: entity = organization_role	
items	type: entity (defined by organization_item_select)	SET[1:?]

Preprocessor Recommendations: For applied_organization_assignments the potential organization_roles 'id owner' and 'creator' are recommended.

Postprocessor Recommendations: Postprocessors should support the assignment of the above roles to products, product_definition_formations and product_definitions (either related to parts or managed documents).

Related Entities: There are no specific related entities.

13.1.1.3   organization_role

Entity represents the role for the assignment of a person_and_organization to product data.

Attributes
- The *name* attribute is the label by which the organization_role is known.
- The *description* attribute characterizes the role with further descriptive text.

ENTITY organization_ role	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	DERIVED

Preprocessor Recommendations: Role.name should be 'creator' or 'id owner' where applicable.

Postprocessor Recommendations: Postprocessors should at least support the above role names. Postprocessors should expect that some existing implementations might export files that use other role names such as 'design supplier' or 'designer'.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#310=APPLIED_ORGANIZATION_ASSIGNMENT(#320,#330,(#270));
#320=ORGANIZATION('ord_id_34','Doc Writers Com',$);
```

Example 52 : exchange file segment for organization

13.1.2 Person and Organization

The PDM Schema specifies information about people through the person entity. A person is identified by an id with other data representing their name and optionally titles which may apply to them. In populating the data, the id must be unique. This is typically not a problem when the person is taken in the context of some specific group such as a company or even country. In these instances, there are typically identifying numbers assigned to people. If the data being assembled is for worldwide consumption, the id must be unique in that domain.

In the PDM Schema, a person always exists in the context of some organization. The connection of people to organizations is accomplished through the person_and_organization entity. This entity is related to certain constructs to identify the people in organizations responsible for them and how they are responsible. This is specified through the applied_person_and_organization_assignment, which relates a person and organization in some role to an entity. The role is established in the name attribute of the person_and_organization_role entity. The person_and_organization_role and person_and_organization entity may have a description associated with it through the entity description_attribute and its attribute_value attribute. The person_and_organization entity may have a name associated with it through the entity name_attribute and its attribute_value attribute. This name describes the relationship of the person to the organization.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements for person and organization are illustrated in Diagram 54.
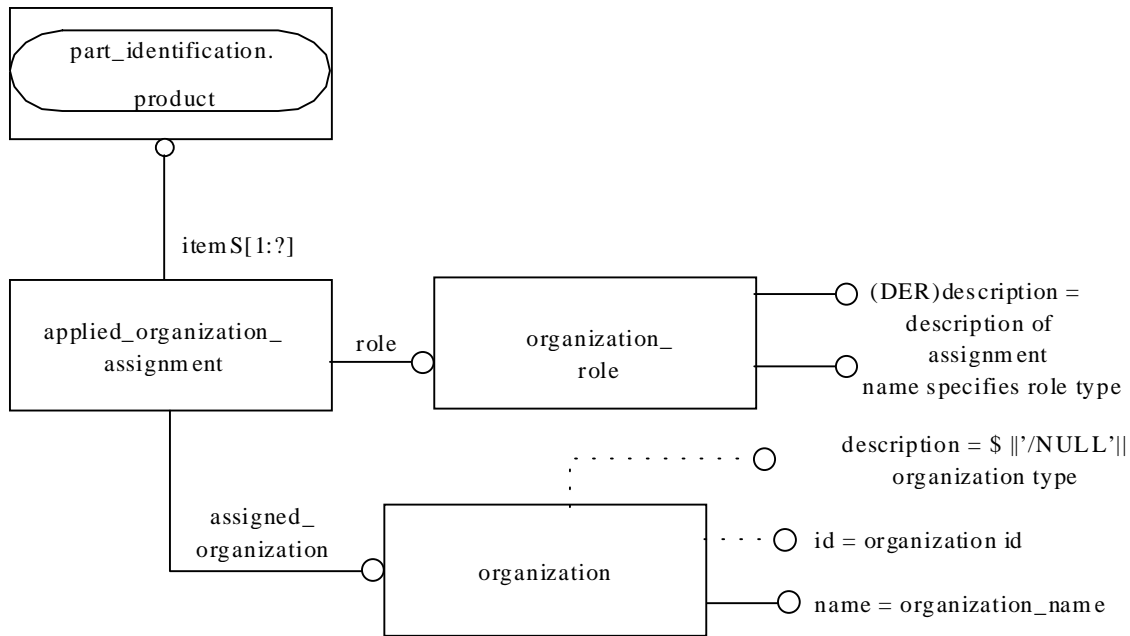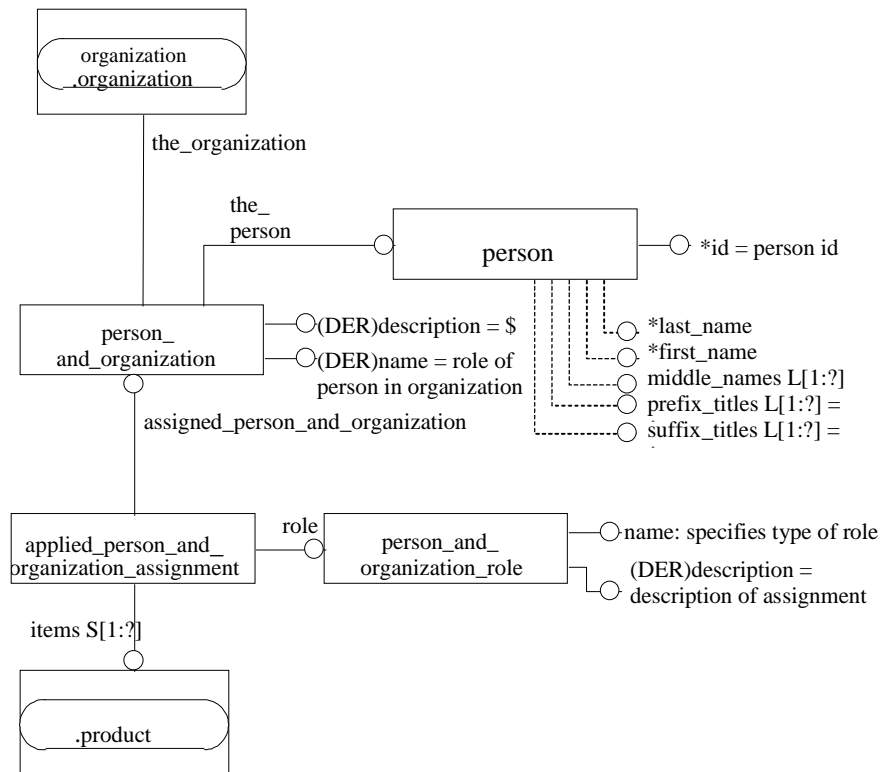
Diagram 54: Person and Organization Instance Diagram

13.1.2.1   person

Entity represents the identification of an individual. In the PDM Schema, a person always exists in the context of some organization.

Attributes

- The *id* attribute distinguishes the person.
- The *last_name* attribute has the last name of the person.
- The *first_name* attribute has the first name of the person.
- The *middle_names* attribute is a list of strings representing the middle names.
- The *prefix_titles* attribute is a list of text which specify the person's social and/or professional standing and appear before his or her names.
- The *suffix_titles* attribute is a list of text, which specifies the person's social and/or professional standing and appear before his or her names relevant for the person name that are used as a suffix.

ENTITY person	Attribute Population	Remarks
id	type: identifier = string	must be unique in the scope of the associated organization(s)
last_name	type: label = string	OPTIONAL it is recommended to instaniate this attribute
first_name	type: label = string	OPTIONAL
middle_names	type: label = string	OPTIONAL SET
prefix_titles	type: label = string	OPTIONAL SET
suffix_titles	type: label = string	OPTIONAL SET

Preprocessor Recommendations: All preprocessors should provide values for at least the last_name and first_name attributes for the person entity in order to provide a sense of meaning to the id attribute. In cases where uniqueness of the id attribute may be a problem, preprocessors should prefix the id attribute with the organization id (as described in the following section) followed by a comma. For example, if the organization id value were "USA,93699" and the person id were "111111", the actual value of the person id would be "USA,93699,111111".

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.1.2.2   person_and_organization

Entity represents the identification of a person within an organization. In the PDM Schema, a person always exists in the context of some organization. The connection of people to organizations is accomplished through the person_and_organization entity.

Attributes
- The *name* attribute is a label, which describes the role of the person in person_and_organization, i.e., the relationship between the person and the organization.
- The *description* attribute characterizes the person_and_organization.
- The *the_person* attribute is the person who is related to the organization.
- The *the_organization* attribute is the organization to which the person is related.

ENTITY person_and_organization	Attribute Population	Remarks
name	type label = string	DERIVE the name is established via the entity name_attribute that has a label for the name as first attribute and references the person_and_organization in the second attribute
description	type: text = string	OPTIONAL
the_person	type: entity = person	
the_organization	type: entity = organization	

Preprocessor Recommendations: Description (i.e. the structure for the derived attribute) shall not be instantiated.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.1.2.3   person_and_organization_role

This entity describes the assignment role for the assignment of a person_and_organization to product data.

Attributes
- The *name* attribute indicates the nature of the assignment.
- The *description* attribute characterizes the person_and_organization_role.

ENTITY person_and_organization_role	Attribute Population	Remarks
name	type: label = string	

| description | type: text = string | DERIVED |

Preprocessor Recommendations: See recommendations for organization_role.

Postprocessor Recommendations: See recommendations for organization_role.

Related Entities: There are no specific related entities.

13.1.2.4   applied_person_and_organization_assignment

This entity is a subtype of person_and_organization_assignment, and represents the assignment of a person_and_organization to product data.

REMARK: In the scope of this usage guide, applied_person_and_organization_assignment is used to associate information about responsibility, not to represent an electronic signature.

Attributes

- The *assigned_person_and_organization* attribute references the person_and_organization.
- The *role* attribute describes the nature of the assignment.
- The *items* attribute is a reference to the product data having the person_and_organization assigned.

ENTITY applied_person_and_ organization_assignment	Attribute Population	Remarks
assigned_person_and_organization	type: entity = person_and_organization	
role	type: entity = person_and_organization_role	
items	type: entity = person_and_organization_item select	SET[1:?]

Preprocessor Recommendations: Entity shall be used when the assignment involves an organization as well as a person related to that organization.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#120=APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT(#130,#170,(#70));
#130=PERSON_AND_ORGANIZATION(#150,#140);
#140=ORGANIZATION('org_id','Miller Company',$);
#150=PERSON('p_id','Miller',$,$,$,$);
#155=NAME_ATTRIBUTE('employee',#130)
#310=APPLIED_ORGANIZATION_ASSIGNMENT(#320,#330,(#270));
#320=ORGANIZATION('ord_id_34','Doc Writers Com',$);
```

Example 53 : exchange file segment for person and organization

13.1.3 Address Assignment

The STEP PDM Schema provides for optional assignment of an address to both organizations and persons.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements for organizational address assignment are illustrated in Diagram 55. The mechanism for personal address assignment is similar.
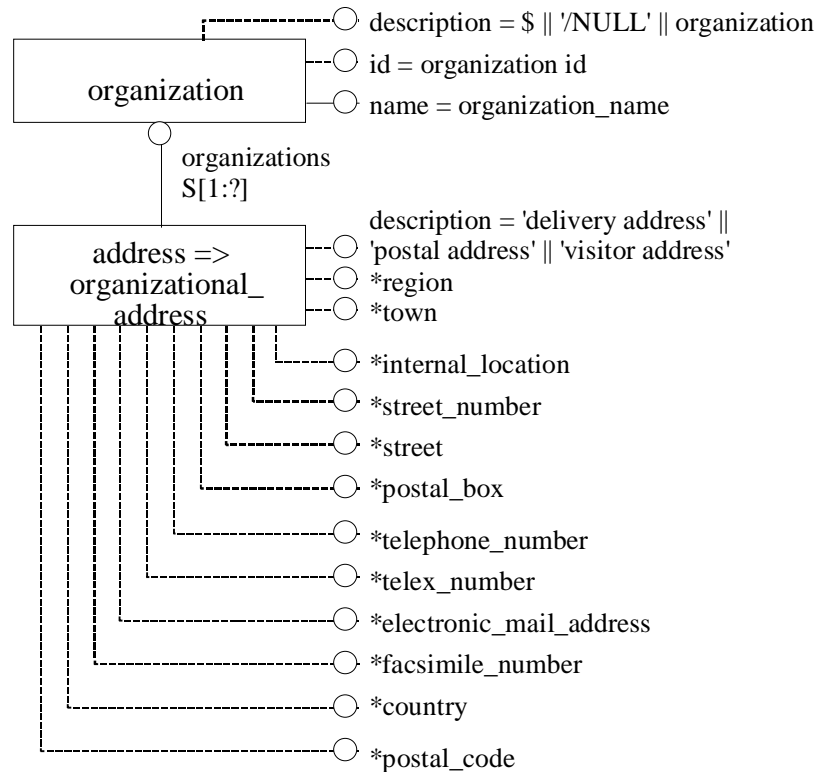
Diagram 55: Address Assignment Instance Diagram (for organization, similar for person)

13.1.3.1   organizational_address

This entity is a subtype of address, which represents the identification of address information relevant for an organization.

Attributes

- The *internal_location* attribute is an organization-defined address for internal mail delivery.
- The *street_number* attribute the number of a location on a street.
- The *street* attribute is the name of the street.
- The *postal_box* attribute is the number of a postal box.
- The *town* attribute is the name of a town.
- The *region* attribute is the name of a region.
- The *postal_code* attribute is the code the is used by the country's postal service.
- The *country* attribute is the name of the country.
- The *facsimile_number* attribute is the number where facsimilies may be received.
- The *electronic_mail_address* attribute is the address where electronic mail messages may be received.
- The *telephone_number* is the number at which telephone calls may be received.
- The *telex_number* attribute.is the number at which telex messages may be received.
- The *description* attribute characterizes the type address to distinguish between delivery, postal and visitor address.
- The *organizations* attribute is a reference to the organization at this address.

ENTITY organizational_address	Attribute Population	Remarks
internal_location		OPTIONAL
street_number		OPTIONAL
street		OPTIONAL
postal_box		OPTIONAL
town		OPTIONAL
region		OPTIONAL
postal_code		OPTIONAL
country		OPTIONAL
facsimile_number		OPTIONAL
telephone_number		OPTIONAL
electronic_mail_address		OPTIONAL
telex_number		OPTIONAL
description		OPTIONAL
organizations		SET[1:?]

Preprocessor Recommendations: The description attribute shall be instantiated with either 'delivery address', 'postal address' or 'visitor address'. For each of these categories, at most one address should be instantiated related to organization. The delivery_address specifies the address where goods are delivered. The postal address specifies the address where letter mail is delivered. A visitor address specifies the address where the organization receives visitors.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.1.3.2   personal_address

Entity a subtype of address, represents the identification of address information to be applied to a person.

Attributes

- The *internal_location* attribute is an organization-defined address for internal mail delivery.
- The *street_number* attribute the number of a location on a street.
- The *street* attribute is the name of the street.
- The *postal_box* attribute is the number of a postal box.
- The *town* attribute is the name of a town.
- The *region* attribute is the name of a region.
- The *postal_code* attribute is the code the is used by the country's postal service.
- The *country* attribute is the name of the country.
- The *facsimile_number* attribute is the number where facsimilies may be received.
- The *electronic_mail_address* attribute is the address where electronic mail messages may be received.
- The *telephone_number* is the number at which telephone calls may be received.
- The *telex_number* attribute.is the number at which telex messages may be received.
- The *description* attribute.characterises the type of address.
- The *people* attribute lists the people that this location.

ENTITY personal_address	Attribute Population	Remarks
internal_location		OPTIONAL
street_number		OPTIONAL
street		OPTIONAL
postal_box		OPTIONAL
town		OPTIONAL

region		OPTIONAL
postal_code		OPTIONAL
country		OPTIONAL
facsimile_number		OPTIONAL
telephone_number		OPTIONAL
electronic_mail_address		OPTIONAL
telex_number		OPTIONAL
description		OPTIONAL
people		SET[1:?]

Preprocessor Recommendations: At least one of the optional attributes shall be instantiated.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#150=PERSON('p_id','Miller',$,$,$,$);
#155=NAME_ATTRIBUTE('employee',#130)
#160=PERSONAL_ADDRESS('office 321',$,$,$,$,$,$,$,'423 -21','423-
33','miller@my-com.com',$,$,'coordinates of Mr Miller');
#320=ORGANIZATION('ord_id_34','Doc Writers Com',$);
#340=ORGANIZATIONAL_ADDRESS($,'14a','E street','P.O 321','My
Town','Nice Region','postal code','My Country','345 - 21','345 -
31','info@company',$,$,$);
```
Example 54 : exchange file segment for address assignment

13.1.4 Recommendations for the assignment of person and organization

Person and organization data considered in this section is information about the responsibilities of persons and organizations in respect to product data. The entities for the assignments are person_and_organization_assignment and organization_assignment. The assignment entity has a role attribute that contains information about the meaning of the assignment. Of the variety of possible items to which the assignments of person and organizational data can be made, Table 2gives the recommendations for scenarios and roles that should be supported.

person_-organization_assignment	product (as part)	product_-definition_-formation (as part version)	product_-definition (as view on part version)	product (as document)	product_-definition_-formation (as document version)
creator		**X**	**X**		**X**
id owner	**X**				**X**	

Table 2: Recommended Assignment of Person/Organization to Product Data

The interpretations of the roles are:

- the role 'creator' indicates that the referenced object has been created (i.e., originally defined) by the organization (or person_and_organization),
- the role 'id owner' describes the person or organization in which the id of the referenced object is valid.

In a given application scenario additional roles might be negotiated. Where applicable the following values may be used:

- 'custodian': The assigned Person or Organization is responsible for the continued existence and integrity of the referenced object;
- 'customer': The assigned Person or Organization acts as a purchaser or consumer of the referenced object;
- 'design supplier': The assigned Person or Organization is the one who is responsible for delivery or the data describing the referenced object;
- 'editor': The assigned Person or Organization is responsible for making any changes to any attribute of the referenced object;
- 'location': The assigned Organization is the place where the referenced object can be found or where it takes place;
- 'manufacturer': The assigned Person or Organization is the one who produces the actual (physical) object;
- 'owner': The assigned Person or Organization owns the referenced object, and has final say over its disposition and any changes to it;
- 'supplier': The assigned Person or Organization is the one who delivers the actual (physical) object (e.g., a dealer);
- 'wholesaler': The assigned Person or Organization is the one who is in the sales chain between the manufacturer and the supplier.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

A complete instantiation example for person and organization assignments is given in Example 55.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('pdm_schema data', 'pdm_schema'), '2;1');
FILE_NAME('C:\document_xample.stp', '1999-5-20 T12:39:1', (''), (), '',
    '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
/*   Entities #10 - #20 define part master data                   */
/*   To the product_definition #90 is a life cycle specific view on */
/*  that part to which organization and person are assigned in the  */
/*  role of the id owner                                            */
#10=APPLICATION_CONTEXT('');
#20=APPLICATION_PROTOCOL_DEFINITION('version
1.1','pdm_schema',1998,#10);
#30=PRODUCT_DEFINITION_CONTEXT_ROLE('',$);
#40=APPLICATION_CONTEXT('mechanical design');
#50=PRODUCT_DEFINITION_CONTEXT('part definition',#40,'design');
#60=PRODUCT_CONTEXT('',#10,'');
#70=PRODUCT('MP-03-2','my part',$,(#60));
#80=PRODUCT_DEFINITION_FORMATION('03','3rd modification',#70);
#90=PRODUCT_DEFINITION('/NULL',$,#80,#50);
#100=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#90,#50,#30);
#110=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#70));

/* Entities #120 to #170 model the asignment of a person within an */
/* organization to the part. The role of this person/organization  */
/* is that of 'id owner'                                           */
#120=APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT(#130,#170,(#70));
#130=PERSON_AND_ORGANIZATION(#150,#140);
#140=ORGANIZATION('org_id','Miller Company',$);
```

```
#150=PERSON('p_id','Miller',$,$,$,$);
#155=NAME_ATTRIBUTE('employee',#130)
#160=PERSONAL_ADDRESS('office 321',$,$,$,$,$,$,$,'423 -21','423-
33','miller@my-com.com',$,$,'coordinates of Mr Miller');
#170=PERSON_AND_ORGANIZATION_ROLE('id owner');

/*   Entities #240 to #300 model a managed document (document as
product)  */
/*   The document gets an assigned organization in the role of
'creation'  */

#240=APPLICATION_CONTEXT('');
#250=PRODUCT_DEFINITION_CONTEXT('digital document definition',#240,'');
#260=PRODUCT_CONTEXT('',#240,'');
#270=PRODUCT('doc_4711','packaging guide','packaging guideline for
part',(#260));
#280=PRODUCT_DEFINITION_FORMATION('ver3.1','version 3.1',#270);
#290=PRODUCT_DEFINITION('id of digital document','digital document for
representing guideline ver3.1',#280,#250);
#300=PRODUCT_RELATED_PRODUCT_CATEGORY('document',$,(#270));

/* Entities #310 to #340 model the assignment of an organization   */
/* to the document defined via #270. The info transferred with the */
/* assignment is that the organization has created the document    */
#310=APPLIED_ORGANIZATION_ASSIGNMENT(#320,#330,(#270));
#320=ORGANIZATION('ord_id_34','Doc Writers Com',$);
#330=ORGANIZATION_ROLE('creation');
#340=ORGANIZATIONAL_ADDRESS($,'14a','E street','P.O 321','My
Town','Nice Region','postal code','My Country','345 - 21','345 -
31','info@company',$,$,$);
ENDSEC;
END-ISO-10303-21;
```

Example 55: exchange file for complete person and organization .

13.2 Approval

Approving in the PDM Schema is accomplished by establishing an approval entity and relating it to some construct through an applied_approval_assignment. The applied_approval_assignment entity may have a role associated with it through the entity role_association and its related object_role entity to indicate the reason/role of this approval related to the particular element of product data.

Approval may be represented as a simple basic approval (see 13.2.1), or it may represent a more complex approval cycle involving multiple approvers, on different dates/times, and possibly with different status values (see 13.2.2).

In a number of STEP APs, constructs that require an approval are allowed only one approval assignment. This might lead to the misconception that only one person at one date/time can approve something.   This is not the case. The approval constructs actually support an approval cycle (see Section 13.2.2). However, an approval may only need one signature corresponding to the simple basic approval scenario.

The level of an approval is understood to represent the aspect for which the approved object is endorsed. This level represents a "state" for which the approved object requires approval. The approval status indicates the level of acceptance given the object for the specified state.

NOTE : The entity product_definition exists to represent different views on a part version. Each view is characterized by a type and a life cycle stage. Various part view "types" correspond to different states that

a part assumes throughout its life cycle. The decision to represent the "state" of a part as a level of approval on a view definition or an entirely different view definition is dependent upon individual business processes. In the STEP PDM schema, different view definitions are typically characterized by different life cycle stages such as 'design' and 'manufacturing'. Approval levels are various "states" defined within a given life cycle stage view, and are typically dependent upon the approval cycles within an individual business process. The "gates" associated with each defined "state" are described by the approval status values leading towards the status 'approved' indicating achievement of the particular level, or "state".

13.2.1 Basic Approval

For basic approval, an approval needs only one signature and does not represent an approval cycle. There is only one approval_person_organization, one approval_status, and one approval_date_time.

It is recommended that the approval entity have associated approval_person_organization to specify the approver of the approval. The approval_person_organization entity has a role associated with it by the entity approval_role. If no more specific information is available, the role 'approver' is recommended.

It is recommended that the approval entity have associated an approval_date_time construct to specify the date when the current approval status was assigned. The approval_date_time may reference either a date or a date_and_time to indicate that the referenced element of product data was approved on the specified date/time.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements for basic approval are illustrated in Diagram 56.
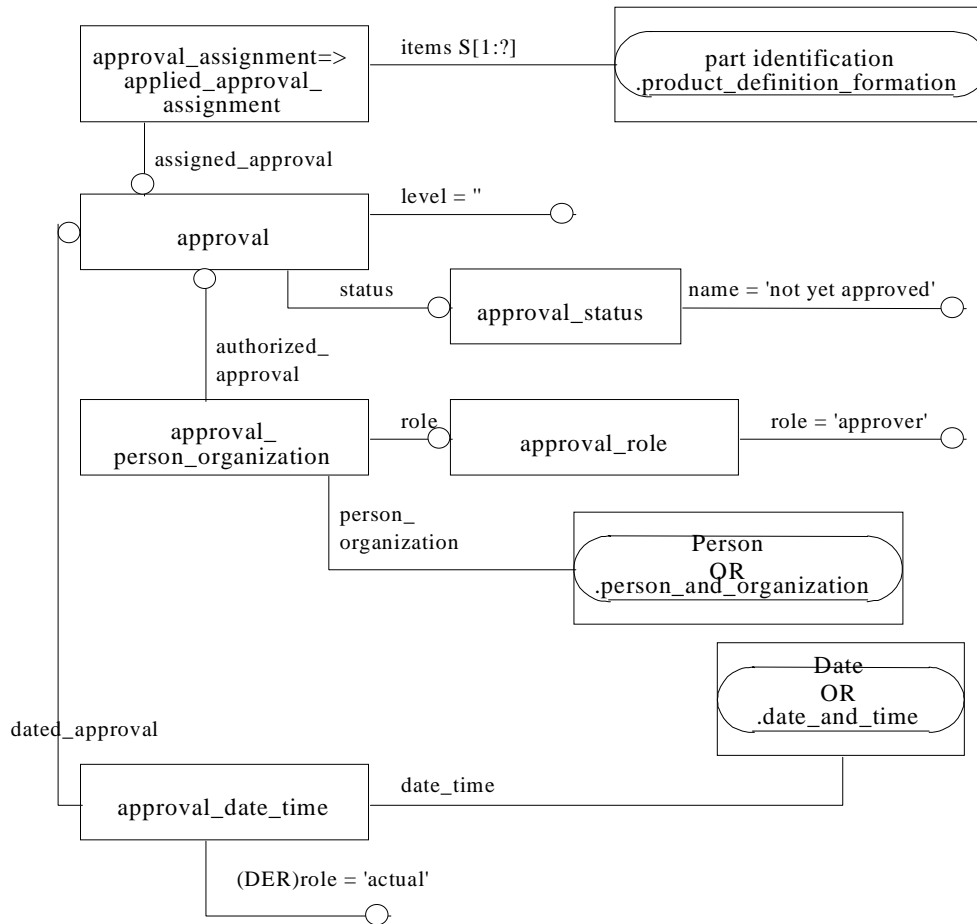
Diagram 56 : Basic Approval Instance Diagram

13.2.1.1   approval

This entity describes the state of acceptance of some product data.

Attributes
- The *status* attribute specifies the judgement made about the product data that is the subject of the approval.
- The *level* attribute describes the type or the level of approval in terms of its usage.

ENTITY approval	Attribute Population	Remarks
status	type: entity = approval_status	
level	type: label	

Preprocessor Recommendations: The approval level attribute is generally agreed to represent the type of approval in terms of its usage. The approved object is understood to be approved for the given purpose described by the value of the attribute level.

Postprocessor Recommendations: None specific.

Related Entities: There are no specific related entities.

13.2.1.2   approval_status

This entity represents a statement made by technical personnel or management personnel whether certain requirements are met.

Attributes
- The *name* attribute describes the terms characterizing the approval_status.

ENTITY approval_status	Attribute Population	Remarks
name	type: label	

Preprocessor Recommendations: The approval_status name attribute is recommended to only be "approved", "not yet approved", "disapproved" or "withdrawn".

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.2.1.3   applied_approval_assignment

This entity is a subtype of approval_assignment, allowing the representation of the assignment of an approval to some product data.

Attributes
- The *assigned_approval* attribute references the approval that is being assigned.
- The *role* attribute describes the nature of the assignment.
- The *items* attribute is a reference to the product data to which the approval is being assigned.

ENTITY applied_approval_assignment	Attribute Population	Remarks
assigned_approval	type: entity = approval	
role	type: entity = object_role	DERIVE
items		SET[1:?]

Preprocessor Recommendations: All preprocessors should use non-defaulted data or user-input data for the values assigned for the approvers and approval date for product_definition_formation entities. The approvers and approval dates can be extrapolated from the version creator and approval data if other appropriate data is unavailable.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.2.1.4   approval_person_organization

This entity specifies who is responsible for the approval.

Attributes
- The ***person_organization*** attribute is a reference to the person or person_and_organization who made the approval.
- The ***authorized_approval*** attribute is a reference to the approval effected by the person and/or organization.
- The *role* attribute describes the nature of the association.

ENTITY approval_person_organization	Attribute Population	Remarks
person_organization	type: SELECT	
authorized_approval	type: entity = approval	
role	type: entity = approval_role	

Preprocessor Recommendations: It is recommended that all approval_person_organization instances have associated applied_date_and_time_assignment entities to provide complete clarity. This is required for the case of multiple sign-offs and approval cycles (see 13.2.2).

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.2.1.5   approval_role

This entity represents the identification of the assignment role for an approval_person_organization.

Attributes
- The *role* attribute describes the terms characterizing the approval_role.
- The *description* attribute specifies the the word or group of words used to refer to the approval_role.

ENTITY approval_role	Attribute Population	Remarks
role	type: label	
description	type: text	DERIVE

The approval_role entity may have a description associated with it through the entity description_attribute and its attribute_value attribute.

Preprocessor Recommendations: If no appropriate data for the approval_role role attribute (why this person and organization is approving) is available it is recommended that this attribute contain the value "approver".

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.2.1.6   approval_date_time

This entity specifies when the approval was given. Year values for dates should always be expressed with four digits, e.g., 1999.

Attributes
- The *date_time* attribute is a reference to the date and/or time when the approval was made.
- The *dated_approval* attribute is a reference to the approval effected by the date and/or time.
- The *role* attribute describes the nature of the association.

ENTITY approval_date_time	Attribute Population	Remarks
date_time	type: SELECT	
dated_approval	type: entity = approval	

role	type: entity = object_role	DERIVE

Preprocessor Recommendations: The approval_date_time.role attribute may contain the value 'actual' to indicate this date/time is the actual (not planned) date that the approval_status related to the dated_approval was assigned.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('pdm_schema data', 'pdm_schema'), '2;1');
FILE_NAME('C:\approval_example.stp', '1999-5-20 T12:39:1', (''), (),
'',
   '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;

/* part version to which an approval is applied */
#10 = PRODUCT_DEFINITION_FORMATION('02', 'lever modified', #20);
#20 = PRODUCT('K01-42051', 'Bicycle Bell RX25B', '', (#30));
#30 = PRODUCT_CONTEXT('', #40, '');
#40 = APPLICATION_CONTEXT('mechanical design');
#50 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#20));
#60=APPLICATION_PROTOCOL_DEFINITION('version1.1','pdm_schema',1998,#40;

/* approval definition / association to product_definition_formation */
#880 = APPROVAL(#870, 'preliminary design');
#870 = APPROVAL_STATUS('not yet approved');
#890 = APPLIED_APPROVAL_ASSIGNMENT(#880, (#10));

/* identification of approver */
#900 = APPROVAL_PERSON_ORGANIZATION(#330, #880, #910);
#910 = APPROVAL_ROLE('approver');
#280 = ORGANIZATION('O-0709', 'Logus GmbH', 'company');
#330 = PERSON_AND_ORGANIZATION(#340, #280);
#340 = PERSON('P-0661', 'Rose', 'Peter', $, $, $);
#350 = NAME_ATTRIBUTE('employee', #330);

/* definition of actual approval date (when status was set) */
#940 = APPROVAL_DATE_TIME(#970,#880);
#950 = ROLE_ASSOCIATION(#960, #940);
#960 = OBJECT_ROLE('actual', $);
#970 = DATE_AND_TIME(#980, #990);
#980 = CALENDAR_DATE(1999, 14, 5);
#990 = LOCAL_TIME(0, 0, 0.0E+000, #1000);
#1000 = COORDINATED_UNIVERSAL_TIME_OFFSET(0, $, .EXACT.);
ENDSEC;
END-ISO-10303-21;
```

Example 56: exchange file segment for basic approval

13.2.2 Approval Cycles and Multiple Sign-off Scenarios

A number of STEP APs have constructs that require an approval, but are allowed only one approval assignment. This might lead to the misconception that only one person at one date/time can approve something. This is not the case. The approval constructs actually support an approval cycle. An approval cycle typically requires multiple signatures to achieve a given status for the overall approval.

Representation of an approval cycle may be done in one of two ways in the PDM Schema, depending upon the specific requirements of the use scenario:

- Single approval with multiple person/organizations,
- Multiple approvals with hierarchical relationships.

If multiple people/organizations are required to sign off in an approval cycle, but only a single status value is maintained for the overall approval, then the single approval with multiple person organization approach is recommended. If individual, possibly different, status values are necessary for each of the multiple sign-offs required in an approval cycle, then the multiple approval with hierarchical relationship structure is recommended.

In all scenarios the approval_date_time records the date/time that the status of an approval was last changed. Year values for dates should always be expressed with four digits, e.g., 1999. The approval_date_time entity may have a role associated it through the entity role_association and its related object_role entity.

The single approval with multiple person/organization type of approval cycle requires multiple signatures but maintains a single approval status. This explains the need for the approval_status of 'not yet approved' - until all signatures are available to switch this status to 'approved'. Each individual sign off should have an associated date/time, through the use of a date_assignment (or date_and_time_assignment) applied directly to the approval_person_organization, to represent the sign off date with the date_time_role of 'sign off'. As always, the approval_date_time indicates when the status of the overall approval was last changed.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements for a cycle involving a single approval with multiple sign-offs are illustrated in Diagram 57. Although a date_time_assignment is used in this example to illustrate a date_and_time, a date_assignment can also be used if only a date without a time specification is required.
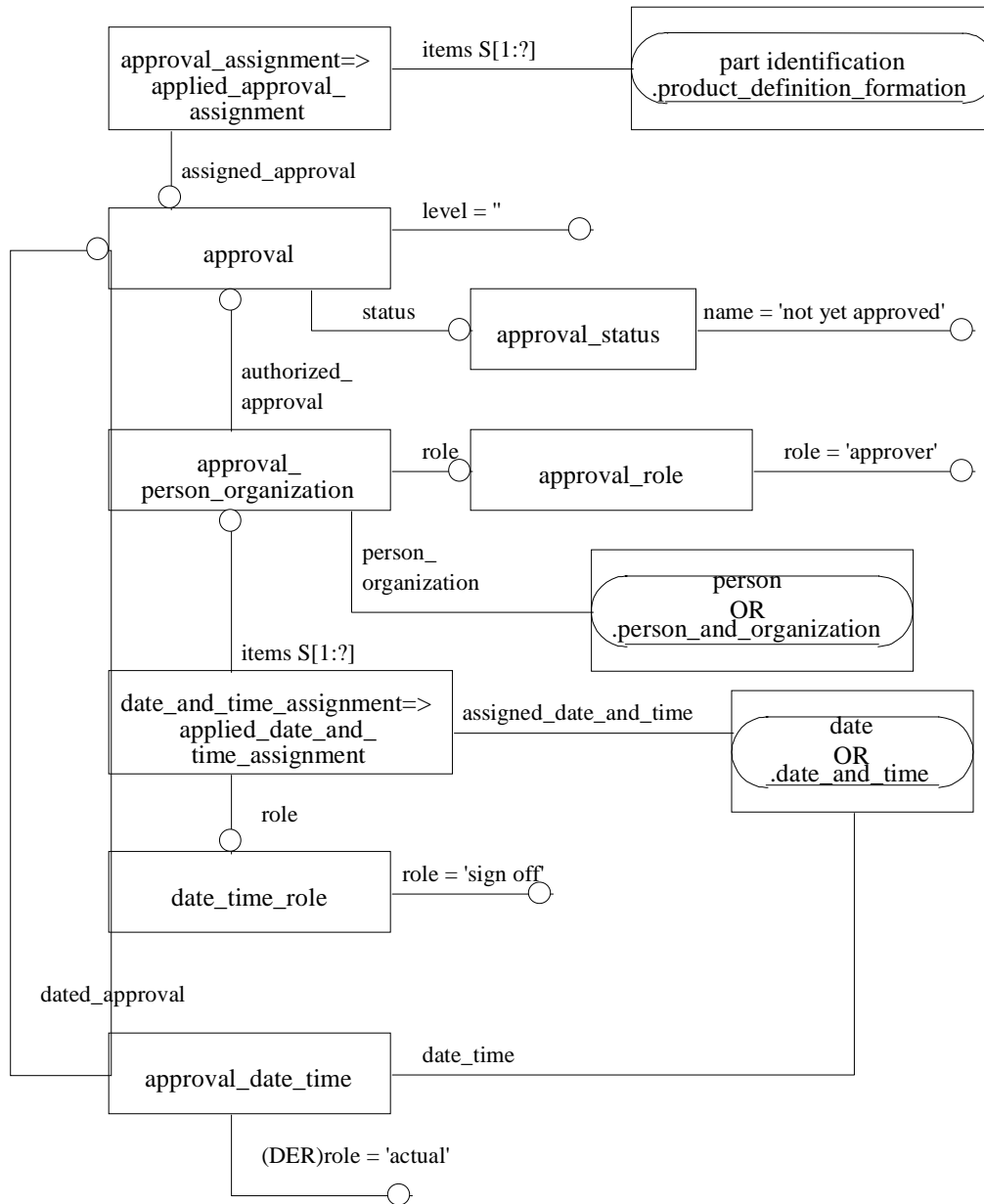
Diagram 57: Single Approval Cycle Instance Diagram

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('pdm_schema data', 'pdm_schema'), '2;1');
FILE_NAME('C:\approval_example.stp', '1999-5-20 T12:39:1', (''), (),
'',
    '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;

/* part version to which an approval is applied */
```

```
#10 = PRODUCT_DEFINITION_FORMATION('02', 'lever modified', #20);
#20 = PRODUCT('K01-42051', 'Bicycle Bell RX25B', '', (#30));
#30 = PRODUCT_CONTEXT('', #40, '');
#40 = APPLICATION_CONTEXT('mechanical design');
#50 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#20));
#60=APPLICATION_PROTOCOL_DEFINITION('version1.1','pdm_schema',1998,#40;

/* approval definition / association to product_definition_formation */
#880 = APPROVAL(#870, '');
#870 = APPROVAL_STATUS('not yet approved');
#890 = APPLIED_APPROVAL_ASSIGNMENT(#880, (#10));

/* definition of overall approval date (when status was set) */
#940 = APPROVAL_DATE_TIME(#970,#880);
#950 = ROLE_ASSOCIATION(#960, #940);
#960 = OBJECT_ROLE('actual', $);
#970 = DATE_AND_TIME(#980, #990);
#980 = CALENDAR_DATE(1999, 14, 5);
#990 = LOCAL_TIME(0, 0, 0.0E+000, #1000);
#1000 = COORDINATED_UNIVERSAL_TIME_OFFSET(0, $, .EXACT.);

/* identification of an approver (one of possibly many) */
#900 = APPROVAL_PERSON_ORGANIZATION(#330, #880, #910);
#910 = APPROVAL_ROLE('approver');
#330 = PERSON_AND_ORGANIZATION(#340, #280);
#280 = ORGANIZATION('O-0709', 'Logus GmbH', 'company');
#340 = PERSON('P-0661', 'Rose', 'Peter', $, $, $);
#350 = NAME_ATTRIBUTE('employee', #330);
/* individual approver sign-off date */
#920 = APPLIED_DATE_AND_TIME_ASSIGNMENT(#1070, #930, (#900));
#930 = DATE_TIME_ROLE('sign off');
#1070 = DATE_AND_TIME(#1080, #1090);
#1080 = CALENDAR_DATE(1999, 12, 5);
#1090 = LOCAL_TIME(0, 0, 0.0E+000, #1000);

ENDSEC;
END-ISO-10303-21;
```

Example 57 : exchange file segment for single approval cycle

The multiple approval with hierarchical relationship type of approval cycle requires multiple signatures, each with individual status values, to achieve a given status value of the overall approval. In this case, each sign-off is itself an approval – a sub-approval of the overall approval. Thus the overall approval is hierarchically broken down into component sub-approvals, each with an individual status, date/time, and person/organization.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements for a cycle involving a multiple approvals with hierarchical approval relationships are illustrated in Diagram 58.
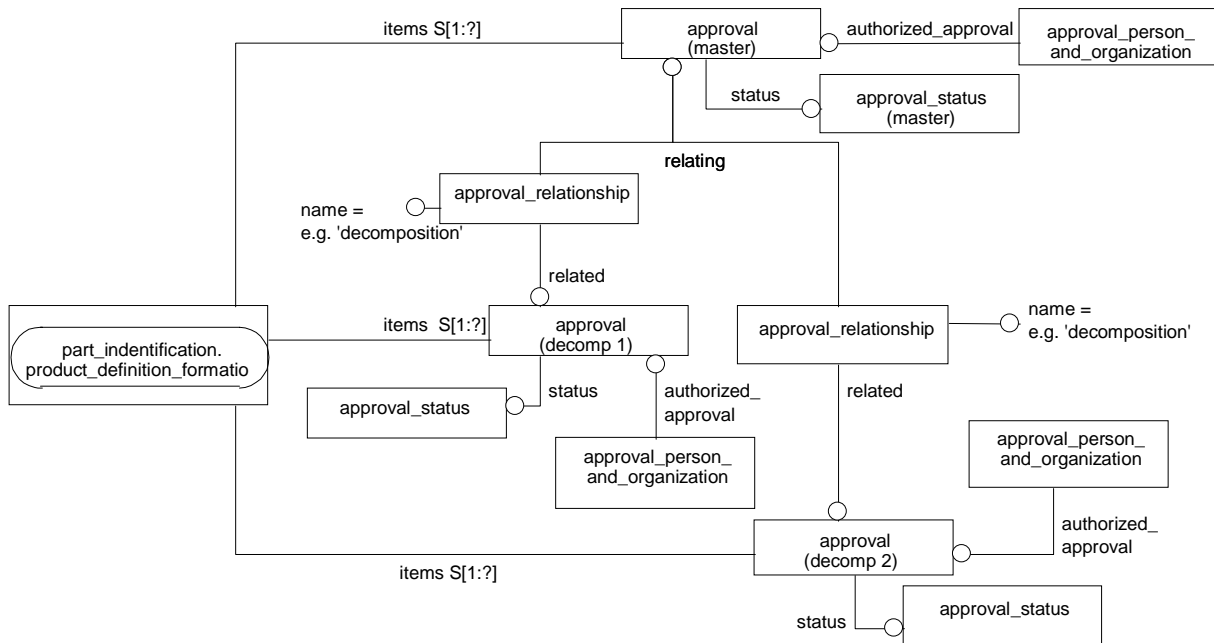
Diagram 58 : Multiple Approval Cycle Instance Diagram

13.2.2.1   approval_relationship

An approval_relationship is a relationship between two approvals, i.e. approval objects. The meaning of the approval relationship is given by its *name* attribute. An approval_relationship may be used to define an approval as part of some higher level approval, or to express some dependency such as sequential relationship in time, between two approval objects.

Attributes
- The *name* attribute specifies the meaning of the relationship.
- The ***description*** attribute specifies additional information text that characterizes the approval_relationship.
- The *relating_approval* specifies the first approval, which is related by the approval_relationship. The relating_approval usually identifies the approval the definition of the related_approval is based on, e.g. derived from or dependent on. The semantics of this attribute should be defined by the name attribute.
- The *related_approval* specifies the second approval related by the approval_relationship. The related_approval usually identifies the approval which is based on the definition of the relating_approval. The semantics of this attribute should be defined by the name attribute.

ENTITY approval_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	OPTIONAL
relating_approval	type: entity = approval	
related_approval	type: entity = approval	

Preprocessor Recommendations: The name attribute is used to specify the type of the approval_relationship and therefore should be populated with predefined values that both sides of the exchange have a common understanding of. The value '**decomposition**' for the name attribute should be used **to indicate that the related_approval** is one of potentially more components into which the relating_approval is broken down. The value 'precedence' for the name attribute should be used **to indicate that the related_approval** has higher priority than the relating_approval. The value 'sequence' for the name attribute should be used **to indicate that the** relating_approval shall be

completed before the **related_approval** is given. There are no further standard mappings specified for the name attribute. Additional values must be mutually agreed upon.

There is no standard mapping for the description attribute. The description attribute is optional, but may contain any appropriate/mutually agreed upon string.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''), '2;1');
FILE_NAME('', '14.10.1999, 11:35:31', ('N.N.'), (''),  '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
/* part master data for product g1 (#4                      */
#10 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #20, #30);
#20 = PRODUCT_RELATED_PRODUCT_CATEGORY('Part', '', (#40));
#30 = PRODUCT_RELATED_PRODUCT_CATEGORY('Detail', $, (#40));
#40 = PRODUCT('g1', 'gear box v1', '', (#60));
#50 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#40));
#60 = PRODUCT_CONTEXT('', #70, '');
#70 = APPLICATION_CONTEXT('');
#80 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema', 1998
   , #70);
#90 = PRODUCT_DEFINITION_FORMATION('1', '', #40);


/* #100 is the top-level approval in the c              */
/* It is via the relationships #235 and #375 decomposed in tw */
/* approval parts. The status in #120 reflects the overall statu */
#100 = APPROVAL(#120, '');
#110 = APPLIED_APPROVAL_ASSIGNMENT(#100, (#90));
#120 = APPROVAL_STATUS('not yet approved');
#130 = ORGANIZATION('dep1', 'departement 1', '');
#160 = APPROVAL_PERSON_ORGANIZATION(#130, #100, #170);
#170 = APPROVAL_ROLE('approver');
#180 = APPLIED_DATE_AND_TIME_ASSIGNMENT(#200, #190, (#160));
#190 = DATE_TIME_ROLE('sign off');
#200 = DATE_AND_TIME(#210, #220);
#210 = CALENDAR_DATE(2000, 1, 1);
#220 = LOCAL_TIME(0, 0, 0.0000E+000, #230);
#230 = COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);


/* #240 is a decomposed part of the top-level approval #10  */
/* This approval has two signing organizations (#270, #32  */
/*attached to it            */
/* The status is reflected in #240                      */
#235 = APPROVAL_RELATIONSHIP('decomposition',$,#100,#250);
#240 = APPROVAL_STATUS('approved');
#250 = APPROVAL(#240, '');
#260 = APPLIED_APPROVAL_ASSIGNMENT(#250, (#90));
```

```
/* #270 is the 1st organization given the approval #240            */
#270 = ORGANIZATION('g1', 'group 1', '');
#280 = APPROVAL_PERSON_ORGANIZATION(#270, #250, #285);
#285 = APPROVAL_ROLE('approver');
#290 = APPLIED_DATE_AND_TIME_ASSIGNMENT(#300, #295, (#280));
#295 = DATE_TIME_ROLE('sign off');
#300 = DATE_AND_TIME(#310, #320);
#310 = CALENDAR_DATE(2000, 1, 1);
#320 = LOCAL_TIME(0, 0, 0.0000E+000, #230);

/* #325 is the 2nd organization given the approval #240           */
#325 = ORGANIZATION('g3', 'group 3', '');
#330 = APPROVAL_PERSON_ORGANIZATION(#325, #250, #335);
#335 = APPROVAL_ROLE('approver');
#340 = APPLIED_DATE_AND_TIME_ASSIGNMENT(#350, #345, (#330));
#345 = DATE_TIME_ROLE('sign off');
#350 = DATE_AND_TIME(#360, #370);
#360 = CALENDAR_DATE(2000, 1, 2);
#370 = LOCAL_TIME(0, 0, 0.0000E+000, #230);

/* #380 is a decomposed approval part of the top-level approval #100*/
/* The status is reflected in #400                              */
#375 = APPROVAL_RELATIONSHIP('decomposition',$,#100,#380);
#380 = APPROVAL(#400, '');
#390 = APPLIED_APPROVAL_ASSIGNMENT(#380, (#90));
#400 = APPROVAL_STATUS('withdrawn');
#405 = ORGANIZATION('g2', 'group 2', '');
#410 = APPROVAL_PERSON_ORGANIZATION(#405, #380, #415);
#415 = APPROVAL_ROLE('approver');
#420 = APPLIED_DATE_AND_TIME_ASSIGNMENT(#430, #425, (#410));
#425 = DATE_TIME_ROLE('sign off');
#430 = DATE_AND_TIME(#440, #450);
#440 = CALENDAR_DATE(2000, 1, 2);
#450 = LOCAL_TIME(0, 0, 0.0000E+000, #230);

ENDSEC;
END-ISO-10303-21;
```

Example 58 : exchange file segment for multiple approval cycle

13.3 Dates, Times, and Event References

13.3.1 Date and Time

The PDM Schema provides mechanisms for assignment of date and time to product data aiming at characterizing when some event or fact occurred. The specification of time is optional, i.e., in many cases only date is required. The applied_date_assignment entity allows for association of date with product data. The purpose of this association is specified by an instance of the entity date_role. As described later in this section, the PDM Schema supports a similar structure for concurrent assignment of date and time.

Dates are represented using the calendar_date entity, i.e., by specifying year, month and day. Time is unambiguously represented using the local_time entity, as it enforces the specification of the time zone, i.e., the offset to the coordinated universal time.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements for date and time are illustrated in Diagram 59.
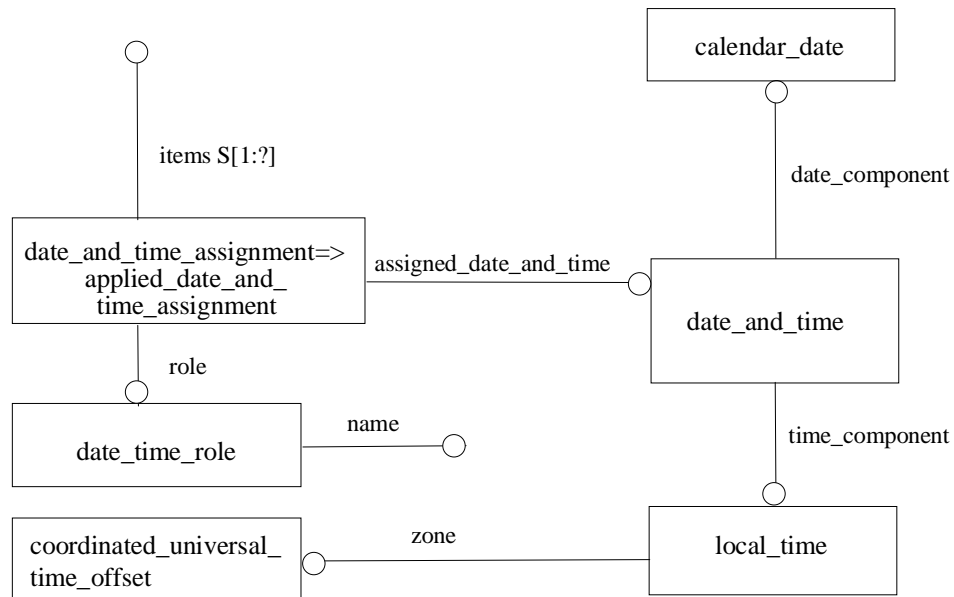
Diagram 59: Assignment of date and time to product data

13.3.1.1   calendar_date

This entity is a subtype of date, which defines it as a day in a month of a year.

Attributes
- The *year_component* attribute specifies the year in which the calendar_date occurs.
- The *day_component* attribute specifies the day element of the calendar_date.
- The *month_component* attribute specifies the month element of the calendar_date.

ENTITY calendar_date	Attribute Population	Remarks
year_component	type: year_number	Year is expressed as a four digits integer
day_component	type: day_in_month_number	Integer between 1 and 31
month_component	type: month_in_year_number	Integer between 1 and 12

Preprocessor Recommendations: The instance shall define a valid calendar date, i.e., it must take into account constraints related to the number of days in a month (e.g., in June the day_component must be an integer between 1 and 30.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.1.2   local_time

This entity represents an instance of time by expressing hour, minute and second in the local time zone and also specifying the offset to the coordinated universal time.

Attributes
- The *hour_component* attribute specifies the number of hours.
- The *minute_component* attribute specifies the number of minutes.
- The *second_component* attribute specifies the number of seconds.
- The *zone* attributes specifies the offset of the local time zone to the coordinated universal time.

ENTITY local_time	Attribute Population	Remarks
hour_component	type: hour_in_day	Integer between 0 and 24, where midnight shall be represented by the value 0.
minute_component	type: minute_in_hour	Integer between 0 and 59. This attribute is optional.
second_component	type: second_in_minute	Real between 0 and 60. This attribute is optional.
zone	type: entity = coordinated_universal_time_ offset	

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.1.3   coordinated_universal_time_offset

This entity specifies allows for relating a time to coordinated universal time by an offset. Coordinated Universal Time (UTC) is the international time standard, which was previously known as Greenwich Meridian Time (GMT).

Attributes
- The *hour_offset* attribute specifies the number of hours by which a time is offset from coordinated universal time.
- The *minute_offset* attribute specifies the number of minutes by which a time is offset from coordinated universal time.
- The *sense* attribute specifies the direction of the offset.

ENTITY coordinated_universal_time_ offset	Attribute Population	Remarks
hour_offset	type: Integer	Value must be between 0 and 12
minute_offset	type: Integer	Value must be between 0 and 59. This attribute is optional
sense	type: ahead_or_behind = ENUMERATION	May assume values ahead, behind or exact

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.1.4   date_and_time

This entity specifies time on a particular day.

Attributes
- The *date_component* attribute specifies the date element.
- The *time_component* attribute specifies the time element.

ENTITY date_and_time	Attribute Population	Remarks
date_component	type: entity = date	In the PDM Schema, it points to an instance of calendar_date.
time_component	type: entity = local_time	

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.1.5   applied_date_assignment

This entity allows for association of date with product data.

Attributes
- The *assigned_date* attribute specifies the date to be assigned to product data.
- The *role* attribute specifies the purpose of the assignment of the date to product data.
- The *items* attribute is a reference to the product data to which the date is being assigned.

ENTITY applied_date_assignment	Attribute Population	Remarks
assigned_date	type: entity = date	In the PDM Schema, it points to an instance of calendar_date.
role	type: entity = date_role	
items	type: date_item = SELECT	SET [1:?]

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.1.6   date_role

This entity specifies a role for an applied_date_assignment.

Attributes
- The *name* attribute defines a label by which the date_role is known.
- The *description* attribute specifies text that characterizes the date_role.

ENTITY date_role	Attribute Population	Remarks
name	type: label	a string value
description	type: text	DERIVED

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.1.7   applied_date_and_time_assignment

This entity allows for association of date and time with product data.

Attributes
- The *assigned_date_and_time* attribute specifies the date_and_time to be assigned to product data.
- The *role* attribute specifies the purpose of the assignment of the date_and_time to product data.
- The *items* attribute is a reference to the product data to which the date_and_time is being assigned.

ENTITY applied_date_assignment	Attribute Population	Remarks
assigned_date_and_time	type: entity = date_and_time	In the PDM Schema, it points to an instance of calendar_date.
role	type: entity = date_time_role	
items	type: date_item = SELECT	SET [1:?]

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.1.8   date_time_role

This entity specifies a role for an applied_date_and_time_assignment.

Attributes
- The *name* attribute defines a label by which the date_role is known.
- The *description* attribute specifies text that characterizes the date_role.

ENTITY date_role	Attribute Population	Remarks
name	type: label	a string value
description	type: text	DERIVED

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#10 = PRODUCT_DEFINITION_FORMATION('02', 'lever modified', #20);
#20 = PRODUCT('K01-42051', 'Bicycle Bell RX25B', '', (#30));
#30 = PRODUCT_CONTEXT('', #40, '');
#40 = APPLICATION_CONTEXT('mechanical design');
#50 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#20));
.
.
.
#920 = APPLIED_DATE_AND_TIME_ASSIGNMENT(#970, #930, (#10));
#930 = DATE_TIME_ROLE('creation date');

/* definition of date and time */
#970 = DATE_AND_TIME(#980, #990);
#980 = CALENDAR_DATE(1999, 14, 5);
```

```
#990 = LOCAL_TIME(0, 0, 0.0E+000, #1000);
#1000 = COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
```

Example 59 : exchange file segment for date and time assignment

13.3.2 Event Reference

In the PDM Schema, events are characterized by instances of the event_occurrence entity or its subtype relative_event_occurrence.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes required to support the requirements for event occurrence assignment are illustrated in Diagram 60. A Part 21 file segment using event reference is shown in Example 60.
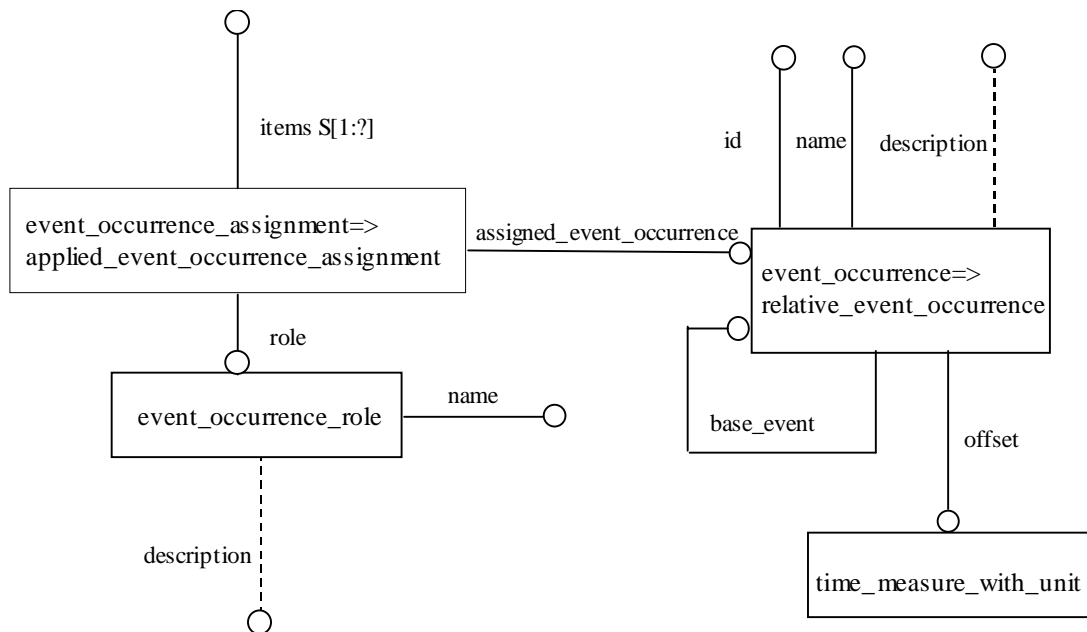
Diagram 60 : Event Occurrence Instance Diagram

13.3.2.1   event_occurrence

This entity describes aspects relevant to characterizing a state transition.

Attributes
- The *id* attribute identifies the event_occurrence.
- The *name* attribute provides the word or group of words by which the event_occurrence is referred.
- The *description* attribute provides additional descriptive information about the event_occurrence.

ENTITY event_occurrence	Attribute Population	Remarks
id	type: identifier	
name	type: label	
description	type: text	OPTIONAL

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: Event_occurrence may be used to specify the limits in dated_effectivity (see 14.2.1.4).

13.3.2.2   relative_event_occurrence

This entity is a subtype of event_occurrence that is characterized by an offset to another event_occurrence.

Attributes
- The *base_event* attribute identifies the event_occurrence providing the basis for definition of the relative_event_occurrence.
- The *offset* attribute specifies the amount of time between the base_event and the relative_event_occurrence.

ENTITY relative_event_occurrence	Attribute Population	Remarks
base_event	type: entity = event_occurrence	
offset	type: entity = time_measure_with_unit	

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.2.3   applied_event_occurrence_assignment

This entity allows for association of events with product data.

Attributes
- The *assigned_event_occurrence* attribute specifies the event_occurrence to be assigned to product data.
- The *role* attribute specifies the purpose of the assignment of the event_occurrence to product data.
- The *items* attribute is a reference to the product data to which the event_occurrence is being assigned.

ENTITY applied_event_occurrence_ assignment	Attribute Population	Remarks
assigned_event_occurrence	type: entity = event_occurrence	
role	type: entity = event_occurrence_role	
items	type: date_item = SELECT	SET [1:?] – In the PDM Schema the select type is constrained to the entity organizational_project.

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

13.3.2.4   event_occurrence_role

This entity specifies a role for an event_occurrence_assignment.

Attributes
- The *name* attribute specifies the word or group of words by which the event_occurrence_role is referred.
- The *description* attribute specifies additional descriptive information about the event_occurrence_role.

ENTITY event_occurrence_role	Attribute Population	Remarks
name	type: label	
description	type: text	OPTIONAL

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#1=APPLICATION_CONTEXT('mechanical design');
#220=PRODUCT_CONTEXT('', #1, '');
#230=PRODUCT_DEFINITION_CONTEXT('part definition', #1, 'design');
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#2));
#2=PRODUCT('11000', 'solid cube', 'description for part 11000',
(#220));
#3=PRODUCT_DEFINITION_FORMATION('A', 'version A for solid cube', #2);
#4=PRODUCT_DEFINITION('D1', 'detailed drawing as planned for STEP
conformance testing', #3, #230);

/* start and end events for effectivity */
#910=EVENT_OCCURRENCE('ev1', 'start production', 'lot for customer A');
#910=EVENT_OCCURRENCE('ev2', 'end production', 'lot for customer A');

/* effectivity definition and assignment */
#1040=DATED_EFFECTIVITY('', #910, #970);
#1042=APPLIED_EFFECTIVITY_ASSIGNMENT(#1040, (#3));
#1044=OBJECT_ROLE('actual', $);
#1045=ROLE_ASSOCIATION(#1044, #1042);
```

Example 60 : exchange file segment for event reference

13.4 Security classification

A security classification is the level of confidentiality that is required in order to protect product data against unauthorized usage.

The Instance Model: EXPRESS entities and attributes

the classified object goes here
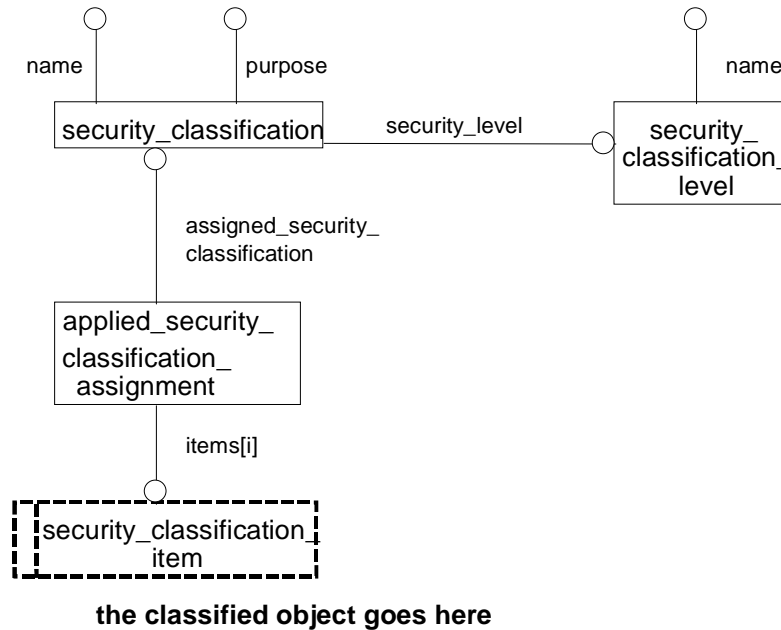
Diagram 61: Part Master with Context Information Instance Diagram

13.4.1.1   security_classification

This entity defines the level of confidentiality that is required for the purpose of product data protection.

Attributes
- The *name* attribute is the label by which the security_classification is known.
- The *purpose* gives an informal description of the intent of the security_classification.
- The *security_level* defines the security_classification_level that specifies the required degree of security.

ENTITY security_classification	Attribute Population	Remarks
name	type : label = string	
purpose	type : text = string	
security_level	type : entity = security_classification_level	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific

*Related Entities***:** None specific

13.4.1.2   security_classification_level

This entity defines a category of security required for product data protection.

Attributes
- The *name* is the label by which the security_classification_level is known.

ENTITY security_classification_level	Attribute Population	Remarks
Name	type: label = string	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities:.

13.4.1.3   applied_security_classification_assignment

This entity is a type of security_classification_assignment that assigns a security_classification to a security_classification_item.

Attributes

- The *assigned_security_classification* gives the security_classification which is to be associated with product data.
- *Items* is the set of items to which the security_classification is assigned. A security classification can be assigned to multiple items with one instance of assigned_security_classification.

ENTITY applied_security_classification_- assignment	Attribute Population	Remarks
assigned_security_classification	type : entity = security_classification	
items	type : entity = security_classification_item	SET [1:?]

Preprocessor Recommendations: None specific

Postprocessor Recommendations: None specific

*Related Entities***:** an assigned_security_classification relates a security_classification to a set to be classified items. The PDM Schema supports the assignment of security classifications to instances of assembly_component_usage, document_file, product_definition and product_definition_formation.

The table below shows the entity types that can be used in the set of items of security classification assignment and the associated semantics.

entity type	semantics of using applied_security_classification with instances of this entity.
assembly_component_usage	classifies the component inclusion in an assemblies.
document_file	classifies a document file
product_definition	classifies the information related to a product_definition, where a product_definition can either model a document representation or a part view.
product_definition_formation	classifies the information related to a product_definition_formation, where a product_definition_formation can either model a document version or a part version.

Table 3 : security classification assignment

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''),'2;1');
FILE_NAME('', '20.08.1999, 13:19:42', (''), (''),
   '', '', '');
FILE_SCHEMA(('PDM_SCHEMA {1.1}'));
ENDSEC;
DATA;
#10 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #20, #30);
#20 = PRODUCT_RELATED_PRODUCT_CATEGORY('Part', '', (#40));
#30 = PRODUCT_RELATED_PRODUCT_CATEGORY('Assembly', $, (#40));
#40 = PRODUCT('as', 'assembly', '', (#60));
#50 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#40, #170));
#60 = PRODUCT_CONTEXT('', #70, '');
#70 = APPLICATION_CONTEXT('');
#80 = APPLICATION_PROTOCOL_DEFINITION('version 1.1', 'pdm_schema', 1998
   , #70);
#90 = PRODUCT_DEFINITION_FORMATION('1', '', #40);
#100 = PRODUCT_DEFINITION('vas', 'design view assembly', #90, #110);
#110 = PRODUCT_DEFINITION_CONTEXT('part definition', #70, 'design');
#140 = PRODUCT_CATEGORY_RELATIONSHIP('', $, #150, #160);
#150 = PRODUCT_RELATED_PRODUCT_CATEGORY('Part', '', (#170));
#160 = PRODUCT_RELATED_PRODUCT_CATEGORY('Detail', $, (#170));
#170 = PRODUCT('p', 'part', '', (#60));
#180 = PRODUCT_DEFINITION_FORMATION('1', '', #170);
#190 = PRODUCT_DEFINITION('vp', 'design view part', #180, #110);
#210 = PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#100, #230, #220);
#220 = PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#230 = PRODUCT_DEFINITION_CONTEXT('assembly definition', #70, '');
#240 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('p_usage_1',
   'single instance usage', '', #100, #190, $);

#250 = SECURITY_CLASSIFICATION('confidentality',
      'classify as confidental',#260);
#260 = SECURITY_CLASSIFICATION_LEVEL('confidental');
#270 =
APPLIED_SECURITY_CLASSIFICATION_ASSIGNMENT(#250,(#180,#190,#240));
ENDSEC;
END-ISO-10303-21;
```

Example 61 : exchange file for security classification

13.5 Certification

The PDM Schema supports the assignment of certification information to parts, component usages and relationships between part versions. The key entity used for this purpose is certification. In the PDM Schema primarily the certification of supplied parts is seen as a use case for certification. Thus the assignment of certification information is restricted to instances of product_definition_ formation_relationship that model a 'supplied part'-relationship.

The Instance Model: EXPRESS entities and attributes

**the objects for which the
certification is applicable**

items

applied_certification_
assignment

assigned_certification

certification ——— kind ——— certification_type

name   purpose

description

certification type is described here
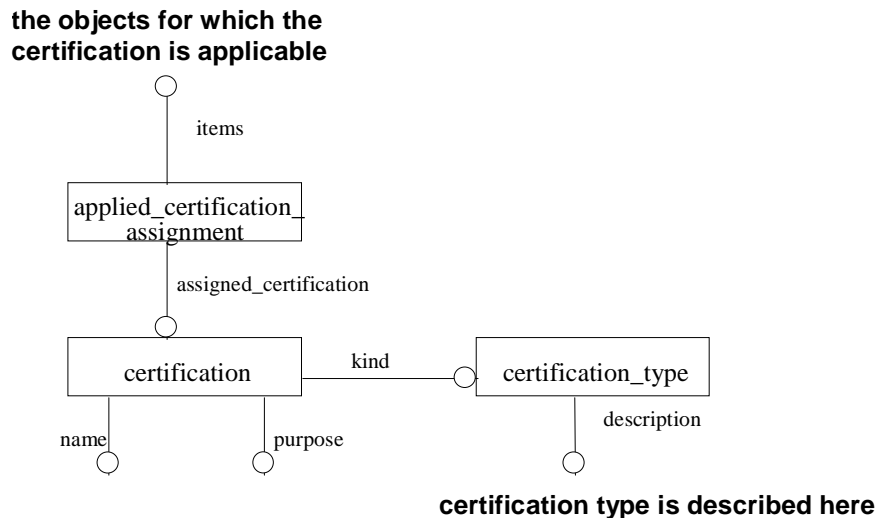
Diagram 62: Certification Instance Diagram

13.5.1.1   certification

This entity is a documentation that asserts facts. Such certification information might for example use ISO 9000 categories.

Attributes
- The *name* attribute defines the label by which a certification is known.
- The *purpose* attribute provides an informal description why the certification is applied.
- The *kind* attribute defines the class of certification that is applied via an instance of certification_type.

ENTITY certification	Attribute Population	Remarks
name	type: label = string	
purpose	type: text = string	
kind	type: entity = certification_type	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: A certification is related to product data information via instances of applied_certification_assignment.

13.5.1.2   applied_certification_assignment

This entity is a type of certification_assignment that assigns a certification to a certification_item.

Attributes
- The *assigned_certification* is the references the relevant instance of the certification entity.

- The *items* attribute defines the set of items to which the certification is assigned. Currently the PDM Schema exclusively supports to use production_definition_formation_relationship as item to be certified.

ENTITY applied_certification_-assignment	Attribute Population	Remarks
assigned_certification	type : entity = certification	
items	type : entity = certification_item	SET[1:?]. In the scope of the PDM Schema this is a set of product_definition_formation_relationship instances

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: Instances of the applied_certification_assignment model establish a relationship between a certification instances and a set of product_definition_formation_relationship instances to which this certification applies.

13.5.1.3   certification_type

This entity gives the kind of certification granted. Such a kind may for example be defined via labels as 'supplier certification' or 'ISO 9000'.

Attributes
- The *description* is a text that characterizes the certification_type.

ENTITY certification_type	Attribute Population	Remarks
description	type: label = string	

Preprocessor Recommendations: None specific.

Postprocessor Recommendations: None specific.

Related Entities: Each instances of certification references an instance of certification_type.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* primary application context for design assembly life cycle */
#10 = APPLICATION_CONTEXT('mechanical design');
#220 = PRODUCT_CONTEXT('', #10, '');
#230 = PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');

/* part type discrimination */
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part',$,(#2, #9));

/* solid cube - supplied part */
#2=PRODUCT('11000','solid 'cube','description for part 11000',(#220));
#3=PRODUCT_DEFINITION_FORMATION('A', 'description of version A for
solid cube,#2);
#4=PRODUCT_DEFINITION('D1',' ',#3,#230);

/* solid box - internal part */
#9=PRODUCT('11111','Solid Box','part 11111 made from raw material
11000',(#220));
```

```
#10=PRODUCT_DEFINITION_FORMATION('A', 'description of version A for
part 11111',#9);
#11=PRODUCT_DEFINITION('D2','',#10,#230);

#16=PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('supplied_item_rel_id','s
upplied item',$,#10,#3);

#17=APPLIED_CERTIFICATION_ASSIGNMENT(#18,#16);
#18=CERTIFICATION('Category A supplied part','purpose is to certify
allowed replacements',#19);
#19=CERTIFICATION_TYPE('supplier certification');
```

Example 62 : exchange file for certification of supplied parts

13.6 Known issues

13.6.1 Organization type

In AP214 ARM, organization has a mandatory type attribute that maps into organization.description. Organization.description is an optional attribute in the PDM Schema. In the PDM Schema, it is not required that organization.description be instantiated.

13.6.2 Role of an date/time assignment for a digital signature

The AP214 ARM says that the role name 'actual' shall be used for a date_time_role specifying the assignment of date/time to approval_person_organization. This role name is already used for approval_date_time to specify the actual date of an approval. In AP203 it is recommended to use 'sign off' as role name in date_time_role. For these reasons, this usage guide recommends use of that value.

13.6.3 Scope in which an approval is valid

Some application protocols provide the capability to specify the scope in which a given approval is valid. A way to support such functionality is to use an applied_organization_assignment that relates an organization – in the role 'scope' – to an approval. The PDM Schema 1.1 does not support this assignment.

13.6.4 Limited security classification capabilities in the PDM Schema

The PDM Schema currently supports exclusively the assignment of security classification level to entities described in the table above. The security classification of parts, managed documents, document files and (external) geometric models are not supported. This contrasts with application protocols as AP214, which provide this additional capability. In addition to this additional capability there is also the requirement to be able to assign organization and date information to security classifications. It should be investigated if a maintenance release of the PDM Schema shall extend the current security classification scope.

14 Configuration and Effectivity Information

The PDM Schema supports the definition of product concepts, which define products from a market or customer oriented viewpoint. As they are offered to the customers, these product concepts often define conceptual 'product models' which are available or delivered to the customer in different configurations or variations.

NOTE: If the number of possible variations of a product concept is too large, it might not be suitable to explicitly specify and manage all existing configurations of that product concept. The members of a product concept might therefore be defined implicitly by identifying the product features that characterize it or are available for it as options. Conditions among those features may be specified to manage dependencies and define the valid product variations for a particular product concept. This implicit definition of the configurations available for a certain product concept is suitable for customer option and variant specification: once all options are specified a single configuration is determined that may be represented explicitly for downstream application. Implicit definition of configurations is not contained in this version of the PDM Schema.

Configuration identification in the PDM Schema is the identification of product concepts and their associated configurations, the composition of which is to be managed. If a configuration of a product concept is implemented by a certain design, i.e. a particular part version, this version can be associated with the configuration and managed using configuration effectivity.

NOTE: Explicit representation of the configurations of a product concept is suitable for management of design as-planned manufacturing configurations, the traditional BOM inputs to manufacturing resource planning. These explicit configurations may also be suitable for management of other activities 'downstream' from the design phase, such as as-built and as-maintained configurations.

Configuration effectivity in the PDM Schema allows attachment of effectivity information to occurrences of component parts in the context of a particular configuration item. This enables specification of the valid use of a part occurrence in the context of a lot, serial number range or time period of a particular product configuration. This controls the constituent parts that are planned to be used for manufacturing end items of a particular product configuration within a dated time period or for a certain lot or serial number range of the end items.

While configuration effectivity is used to define the planned usage of components in the context of a particular product configuration, the PDM Schema also allows assignment of general validity periods to product data that control the usage of these product data independent of any particular life cycle or context.

14.1 Configuration Identification

The configuration identification supports two important concepts to describe the products that are sold by an organization to its customers:

- Product Concept Identification,
- Product Concept Configuration Identification.

Product concept identification supports the representation of an organization's products as they are conceived or offered to the customers. A product as offered to the market can directly correspond to a manufacturable object, or it can be available to the customer in different configurations where each of these configurations defines a manufacturable object. The product concept configuration identification supports the specification of these product configurations and their associated part designs.

14.1.1 Product Concept Identification

The products that are sold by an organization to its customers are often defined as variations or configurations of a common product model, referred to as product concept. This product concept is defined in a market context and can be seen as a logical container for all of its variations. In the PDM Schema, product concept identification is the representation of these product concepts with their related market context.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements to represent product concept identification information are illustrated in Diagram 63 below.
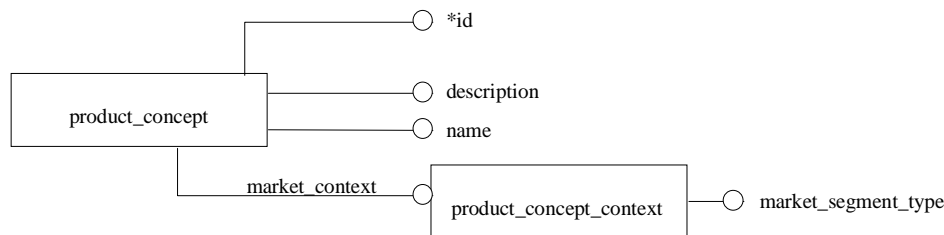
Diagram 63 : Product concept identification Instance Diagram

14.1.1.1   product_concept

A product_concept describes a class of similar products that an organization provides to its customers. It represents the idea of a product as identified by potential or actual customer requirements. Therefore, a product_concept may exist before the parts have been defined that implement it.

Depending on the kind of industry and products, a product_concept might be offered to the customers in one or many different configurations. If exactly one configuration is defined, the product_concept corresponds to a particular part design. If the product concept is offered in different configurations, each of these configurations is a member of the class of products defined by this product_concept.

Attributes
- The *id* attribute specifies the identifier of the product_concept. It is usually assigned by the organization that provides the product belonging to that product_concept to its customers. This identifier must be unique within the given scope.
- The *name* attribute specifies the nomenclature, or common name, by which the product_concept is referred to.
- The *description* attribute specifies additional information about the product_concept.
- The market for the products belonging to a product_concept is further described in the *market_context* attribute. All product_concepts must have a related product_concept_context, which specifies their market context.

ENTITY product_concept	Attribute Population	Remarks
id	type: identifier = string	must be unique
name	type: label = string	
description	type: text = string	OPTIONAL
market_context	type: entity = product_concept_context	reference to the associated product_concept_context (see 14.1.1.2)

Preprocessor Recommendations: There might be several levels of product concepts defined in a company. The description of product concept structures and relationships is not supported by the current PDM Schema version. It is therefore recommended to instantiate the lowest level product_concept that is offered to the customer.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A product_concept represents a conceptual idea of a class of similar products that are offered to a market. No design or manufacturing related product data can be attached to the product_concept directly. The members of that product class are the different configurations that are available for the product_concept. Within the PDM Schema, these configurations are defined explicitly as configuration_items (see 14.1.2.1), each of them being potentially related to the parts that implement the particular configuration. A product_concept will usually be referenced by at least one associated configuration_item specifying a particular product configuration.

In addition, a product_concept may be referenced by the following entity:
- applied_organizational_project_assignment to represent projects associated to the product_concept (see 15.3.1.2).

14.1.1.2   product_concept_context

The product_concept_context entity is a subtype of application_context_element. It defines the market context in which a product_concept is defined and may include information characterizing the potential customers of the products of the associated product_concept. The application domain is identified by the associated application_context entity.

Attributes
- The *market_segment_type* is a label that identifies the kind of consumer preferences associated with a product_concept.

ENTITY product_concept_context	Attribute Population	Remarks
name	type: label = text	Specifies a label by which the product_concept_context is known.
frame_of_reference	type: entity = application_context	see 1.1.2.2
market_segment_type	type: label = text	

Preprocessor Recommendations: It is recommended to instantiate this entity exactly once in the data set. There is no standard mapping for the name and market_segment_type attributes of a product_concept_context. Neither AP203 nor AP214 define requirements for a product_concept_context. It is therefore recommended to use the default values for the entities and attributes not supported by the preprocessor as described in the 'General information' section of this usage guide.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: Related Entities: There are no specific related entities.

14.1.2 Product Concept Configuration Identification

Product configuration identification is the explicit identification of the configurations that exist for a given product concept and the representation of their composition. A product configuration is specified as a configuration_item. Product configurations can be related to appropriate part versions to specify the designs that implement the product configurations through the configuration_design.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the identification of customer oriented product configurations are illustrated in Diagram 64 below. The corresponding STEP exchange file encoding is given in Example 63.
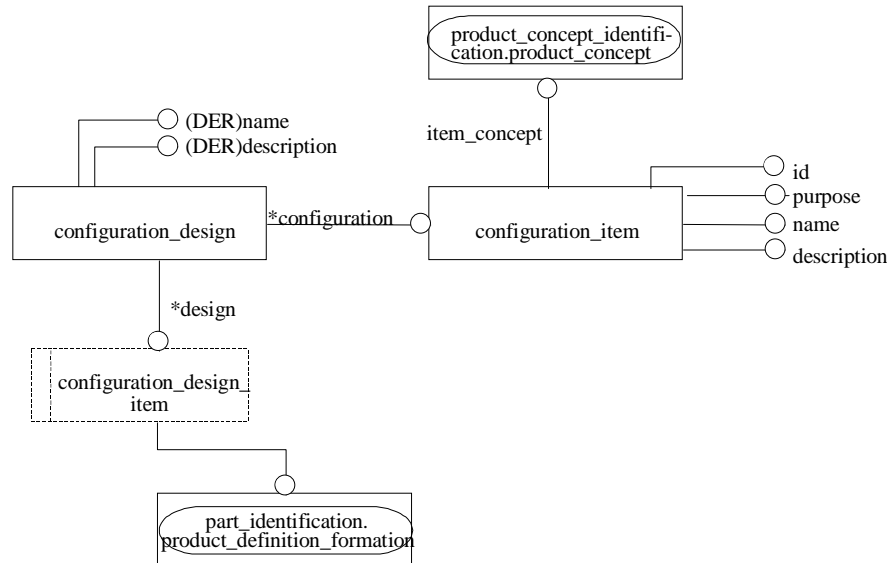
Diagram 64: Product Concept Configuration Identification Instance Diagram

14.1.2.1   configuration_item

The configuration_item entity is a key concept to support explicit configuration management. It represents the identification of a particular configuration, i.e. variation of a product_concept. A configuration_item is defined with respect to the product concept, i.e., the class of similar products of which it is a member.

The configuration_item defines a manufacturable end item, or something that is conceived and expected as such. The association between a configuration_item and a corresponding part design is established using a configuration_design. The valid use of component parts for planned units of manufacturing of a particular configuration_item may be specified using configuration effectivity (see 14.2).

NOTE: Depending on the type of products, and the organization that defines and sells the products, a product_concept may directly correspond to a particular part design, which means that only one variation, i.e. one configuration_item exists for that particular product_concept.

NOTE: ISO 10303-44 allows the configuration_item to be a variation of a product concept, or of any of its discrete portions that is treated as a single unit in the configuration management process. According to this definition, it is possible to associate a configuration_item describing a particular configuration of, for example, an engine to a product concept describing a class of cars. This usage is not recommended.

Attributes
- The *id* attribute specifies an identifier that distinguishes the configuration_item.
- The *name* attribute specifies the word or common name used to refer to the configuration_item.
- The *description* attribute specifies **additional information about** the configuration_item.
- The *item_concept* attribute specifies the product_concept the configuration_item is a member, i.e. a configuration of.
- The *purpose* attribute specifies a descriptive label providing a reason to create the item_concept.

ENTITY configuration_item	Attribute Population	Remarks
id	type: identifier = string identification of view owner	must be unique in relation with a specific product_concept
name	type: label = text	may be 'product identification'
description	type: text = string	OPTIONAL
item_concept	type: entity = product_definition_formation	reference to the product_concept the configuration_item is a configuration of.
purpose	type: label = text	OPTIONAL

Preprocessor Recommendations: There is no standard mapping for the id attribute of configuration_item; however the value should be unique in conjunction with the id attribute of the associated product_concept.

The description attribute is optional, but may contain any appropriate/mutually agreed upon string. If this attribute is not used, it should be assigned the value '$'.

The purpose attribute is optional, but may contain any appropriate/mutually agreed upon string. In AP203 it is used to convey the stage in the life cycle of the product (e.g. Technology and Concept Development, Development and Refinement, Manufacturing and Assembly Validation, and Production) in which a particular version of the design of a configuration_item is being exchanged. If this attribute is not used, it should be assigned the value '$'.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A configuration_item may be referenced by the following entities:
- configuration_design to associate the configuration_item to a corresponding part design. A configuration_item can be established prior to the existence of a corresponding part design, i.e. a configuration_design does not need to exist for a configuration_item.

14.1.2.2   configuration_design

The configuration design entity represents an association between a configuration_item identifying a particular product configuration and a product design intended to implement that item. Thus, a configuration_design entity represents the association of a configuration_item with a product_definition or product_definition_formation to specify that the corresponding design is a solution for a given configuration_item. A given design may be the design for different configuration_items belonging to the same or even different product_concepts.

The configuration_design entity may have at most one name associated with it through the entity name_attribute. If a name is associated, the configuration_design is referenced as the named_item by a name_attribute where the name is stored in the attribute attribute_value. It is not recommended to instantiate this additional value.

The configuration_design entity may have a description associated with it through the entity description_attribute. If a description is associated, the configuration_design is referenced as the described_item by a description_attribute where the description is stored in the attribute attribute_value. It is not recommended to instantiate this additional value.

Attributes
- The ***configuration*** attribute specifies the configuration_item, i.e. the product configuration that the associated design implements.
- The ***design*** attribute specifies the product_definition or product_definition_formation that is a candidate for use in manufacturing actual units of a configuration_item.

ENTITY configuration_design	Attribute Population	Remarks
configuration	type: entity = configuration_item	reference to the configuration_-item for which a design is specified
design	type: entity = configuration_design_item select	reference to the product_definition_formation that is a design for the associated configuration_item
name	DERIVEd attribute type: label = string	at most one name can be associated via the entity name_attribute
description	DERIVEd attribute type: text = string	at most one description can be associated via the entity description_attribute

Preprocessor Recommendations: The design attribute allows specifying a product_definition or product_definition_formation as the design for the associated configuration_item. Both AP203 and AP214 do not foresee product_definitions to be specified as the design in a configuration_design. It is therefore recommended to only reference product_definition_formations from the configuration_design.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A configuration_design may be referenced by the following entities:
- configuration_effectivity to specify which component parts are planned to be used for building actual units of manufacturing of the associated configuration_item. (See 14.2)

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#100=PRODUCT_CONCEPT_CONTEXT('pcc_name1', #1, '');
#1=APPLICATION_CONTEXT('');
#2=PRODUCT_CONCEPT('PC-M01', 'PC model name1', 'PC system', #100);
#5=CONFIGURATION_ITEM('PC-Conf1', 'Base Config Europe', 'PC system
standard configuration for Europe', #2, $);
#6=CONFIGURATION_ITEM('PC-Conf2', 'Base Config US', 'PC system standard
configuration for US', #2, $);
#7=CONFIGURATION_ITEM('PC-Conf3', 'High End Config US', 'High end PC
system for US', #2, $);
#8=APPLICATION_CONTEXT('');
#9=PRODUCT_CONTEXT('', #8, '');
#10=PRODUCT('PC-0023', 'PC system', $, (#9));
#11=PRODUCT_DEFINITION_FORMATION('D', 'housing redesigned', #10);
#12=PRODUCT_DEFINITION_CONTEXT('part definition', #8, 'design');
#13=PRODUCT_DEFINITION('pc_v1', 'design view on base PC', #11, #12);
#14=PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#10));
#15=CONFIGURATION_DESIGN(#5, #11);
#16=CONFIGURATION_DESIGN(#6, #11);
#17=CONFIGURATION_DESIGN(#7, #11);
```

Example 63: exchange file segment for configuration identification

14.2 Configuration Composition Management

Configuration composition management is concerned with the specification of the different product configurations that exist for a given product concept, and the association with product data that is necessary

to build those configurations. This includes the identification of the actual constituents that are to be included in a planned unit of production of a product configuration.

Effectivity is a key concept to control the valid use of product data. The PDM Schema supports the association of effectivity information to different types of product data. Two different effectivity concepts are available:

- configuration effectivity, describing the planned use of part occurrences (i.e. occurrences of parts as sub-assemblies or component parts in some higher level assembly) in the context of a configuration_item defined for a product_concept.
- a more generic effectivity, describing the validity period in which the associated product data may be used independent from any additional context (e.g. life cycle or organization related) that further restricts the applicability of that effectivity (see 14.3).

14.2.1 Configuration effectivity

Configuration effectivity allows control of the constituent parts that should be used to build the physical instances of a product configuration. The composition of the product configurations for planned units of manufacture may be controlled for a given time period, lot, or serial number range. This is managed using dated_effectivity, lot_effectivity, or serial_numbered_effectivity.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements of configuration effectivity are illustrated in Diagram 65 below. The corresponding STEP exchange file encoding is given in Example 64.
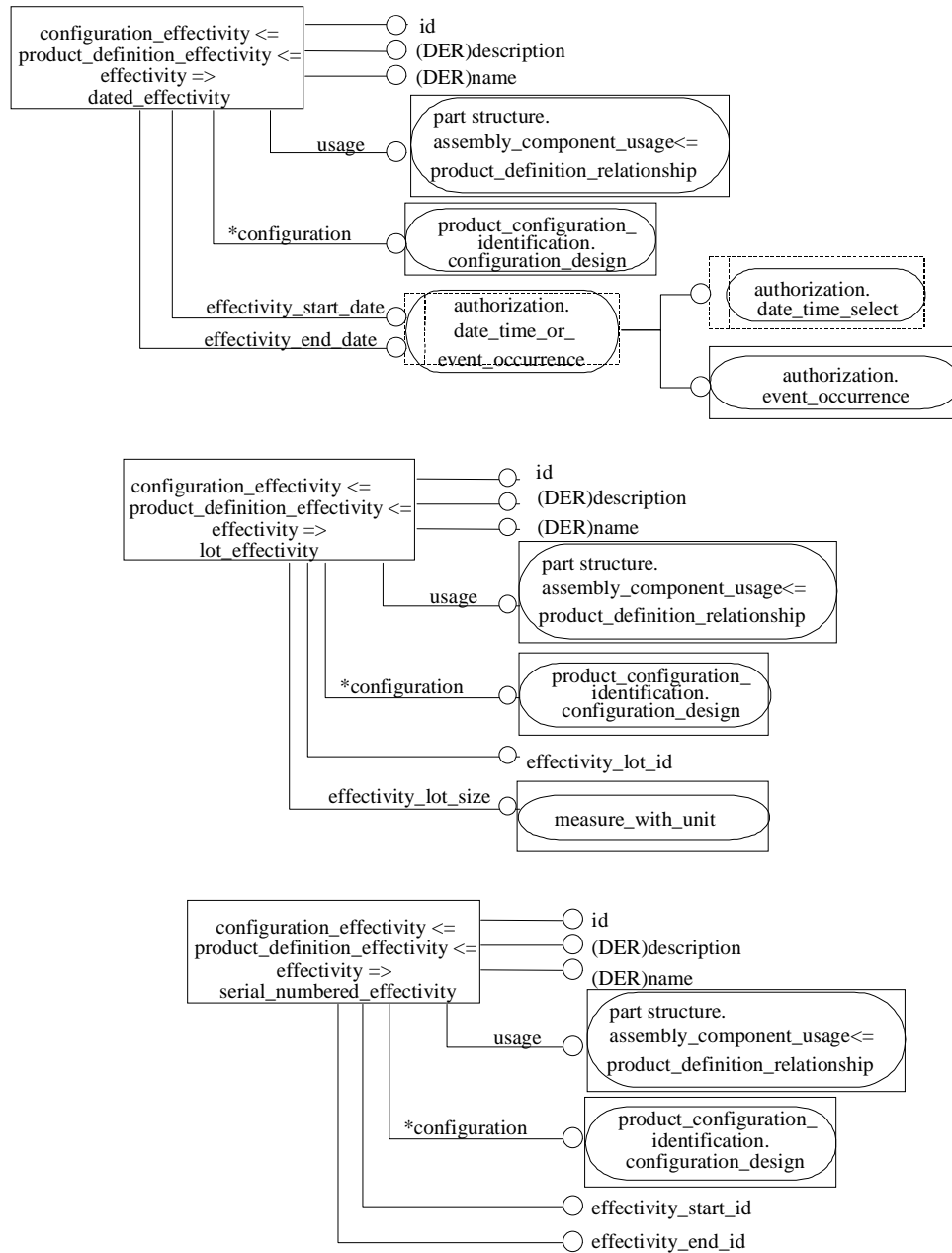
Diagram 65: Configuration Effectivity Instance Diagram

14.2.1.1   effectivity

The effectivity entity supports the specification of a domain of applicability for product data. Effectivity is a generic concept defining the valid use of the product data the effectivity is assigned to.

NOTE: The assignment of effectivities is often done during the approval process, i.e. when releasing product data for the next stage of the development process, it gets effectivity information assigned to define when and in which context these product data may be used.

Within the PDM Schema, there are two areas identified for the usage of the effectivity concept: configuration effectivity and general validity period effectivity. These two areas are reflected by two orthogonal subtype structures defined for the effectivity entity.

One subtype structure contains the effectivity subtypes dated_effectivity, lot_effectivity, serial_numbered_effectivity, and time_interval_based_effectivity, restricting the domain of applicability of the associated product data to a date range, a particular lot or serial number range, or to a time interval respectively. There is a ONEOF constraint between these subtypes, i.e. an effectivity can be at most one of dated_effectivity, lot_effectivity, serial_numbered_effectivity, and time_interval_based_effectivity.

The other subtype structure contains product_definition_effectivity as subtype of effectivity, and configuration_effectivity as subtype of product_definiton_effectivity. The subtypes in the second structure are restricted to apply to a particular usage occurrence of a constituent part in some higher-level assembly.

Attributes
- The *id* attribute specifies the identifier that distinguishes the effectivity.

ENTITY effectivity	Attribute Population	Remarks
id	type: identifier = string	idenitifier *of the effectivity*
name	DERIVEd attribute type: label = string	at most one name can be associated via the entity name_attribute
description	DERIVEd attribute type: text = string	at most one description can be associated via the entity description_attribute

Preprocessor Recommendations: It is recommended to not instantiate the effectivity instance as standalone instance, i.e. an effectivity should always be assigned to some product data either by using the applied_effectivity_assignment or by using the subtype configuration_effectivity, or the effectivity should have a relationship to some other effectivity that is assigned to some product data through effectivity_relationship (see 14.3.1.2) . Furthermore, it is always recommended to instantiate an effectivity as one of its subtypes, unless the definition of the effectivity is based on some other effectivity using effectivity_relationship (see 14.3.1.2) .

An identifier need not to be specified for a particular effectivity. If used, the id value should be unique in conjunction with the product data the effectivity is assigned to, and if present, the context restricting the applicability of the effectivity to a certain usage.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An effectivity may be referenced by the following entities:
- applied_effectivity_assignment to assign the effectivity to the product data it applies to. Applied_effectivity_assignment is only used in general validity period assignment (see 14.3); it may not be used to assign product_definition_effectivity.
- effectivity_relationship to associate two effectivities e.g. to base the definition of one effectivity on the definition of the other, or to describe their dependencies.

14.2.1.2   product_definition_effectivity

The product_definition_effectivity entity is a subtype of effectivity.   It applies to a particular occurrence of a component or sub-assembly part as used within an assembly. It is the identification of a valid use of a

particular product_definition in the context of its participation in a given product_definition_usage. The effectivity applies to a product_definition that is referenced as the related_product_definition by the product_definition_relationship associated to the product_definition_effectivity through the *usage* attribute.

Attributes
- The *usage* attribute specifies the product_definition_relationship defining the product_definition to which the effectivity applies as the related_product_definition, and identifying the usage context for which the effectivity applies.

ENTITY product_definition_effectivity	Attribute Population	Remarks
id	see supertype	
name	see supertype	DERIVEd attribute
description	see supertype	DERIVEd attribute
usage	type: entity = product_definition_relationship	reference to the associated product_definition_relationship

Preprocessor Recommendations: In the context of the PDM Schema, product_definition_effectivity is only used as configuration_effectivity, i.e. product_definition_effectivity should always be instantiated as its subtype configuration_effectivity.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

14.2.1.3   configuration_effectivity

The entity configuration_effectivity is a subtype of product_definition_effectivity that contains information about the planned usage of components in a product configuration. It defines the valid use of a particular product_definition occurrence at a certain position in the assembly structure for a specified product configuration.

The configuration_effectivity entity allows the association of the appropriate versions of constituent parts intended to be used at the defined position in the assembly structure to build a **configuration_item**. In the context of the PDM Schema, the configuration_effectivity always relates to a particular unit of manufacture of the end items of the given configuration_item. The three ways to instantiate configuration_effectivity are:

- *serial_numbered_effectivity*, where the configuration_effectivity defines the valid use of constituent part occurrences for a serial number range of instances of a product configuration to be manufactured;
- *dated_effectivity*, where the configuration_effectivity defines the valid use of constituent part occurrences for a time range based on dates when instances of the product configuration are manufactured;
- *lot_effectivity*, where the configuration_effectivity defines the valid use of constituent part occurrences for a given lot of instances of a product configuration to be manufactured.

Attributes
- The *configuration* attribute identifies the configuration_design that specifies the associated configuration_item for which the configuration_effectivity applies, and the corresponding product_definition_formation which defines the design that implements the configuration_item and thus specifies the upper most node ('entry point') in its product structure.

The particular usage of the constituent part the effectivity applies to is defined as product_definition_usage referenced by the *usage* attribute. The product_definition_usage should be instantiated as an assembly_component_usage or one of its subtypes (see 4.1.1.1). The attribute related_product_definition specifies the definition of a constituent part, and the attribute relating_ product_definition specifies the

definition of the assembly (or sub-assembly) in which the constituent may be used.   . This assembly definition is part of the design that implements the configuration_item for which the configuration_effectivity applies, i.e. it either is a part definition of the product_definition_formation defining the ('complete') design for the configuration_item, or it is an occurrence in the product structure of that complete design, i.e. is related to that in a tree of assembly_component_usage instances.

ENTITY configuration_effectivity	Attribute Population	Remarks
id	see supertype	The combination of the id attribute, the associated product_definition_usage and configuration_design has to uniquely identify the configuration_effectivity.
name	see supertype	DERIVEd attribute
description	see supertype	DERIVEd attribute
usage	see supertype	type is restricted to be a product_definition_usage <= product_definition_relationship
configuration	type: entity = configuration_design	reference to the configuration_design specifying the configuration_item which identifies the context in which the effectivity applies.

Preprocessor Recommendations: A configuration_effectivity should always be instantiated in conjunction, i.e. as complex instance with either dated_effectivity, lot_effectivity, or serial_numbered_effectivity to specify the dated time period, or the lot or serial number range for the manufacturing units of the associated configuration_item, for which the effectivity applies.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A configuration_effectivity may be referenced by the following entity:
- applied_approval_assignment to assign approval information to the configuration_effectivity.

14.2.1.4   dated_effectivity

The dated_effectivity entity is a subtype of effectivity that applies onwards from a point in time, or between two points in time which define the start and end of the dated_effectivity. These points in time may be specified as date or a date and time, or by an event ocurrence.

Attributes
- The *effectivity_start_date* specifies the date_and_time or the event when the effectivity of the associated product data begins to apply. It defines the lower bound of the interval of applicability.
- The *effectivity_end_date* specifies the **date_and_time** or the event when the effectivity of the associated product data ends. It defines the upper bound of the interval of applicability. If a value for this attribute is not defined, the interval of applicability has no upper limit.

ENTITY dated_effectivity	Attribute Population	Remarks
id	see supertype	
name	see supertype	DERIVEd attribute
description	see supertype	DERIVEd attribute
effectivity_start_date	type: entity = date_time_or_event_occurrence	
effectivity_end_date	type: entity =	OPTIONAL

	date_time_or_event_occurrence	

Preprocessor Recommendations: In the context of the PDM Schema, dated_effectivity is used to specify configuration_effectivities, as well as general validity periods of product data. If used to specify the valid use of a part occurrence in the context of a date range for end items of an identified product configuration, dated_effectivity has to be instantiated in conjunction, i.e. as complex instance with configuration_effectivity.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

14.2.1.5   lot_effectivity

The entity **lot_effectivity** is a subtype of **effectivity**. It defines the domain of applicability as a given batch of end items.

Attributes
- The *effectivity_lot_id* attribute specifies a production batch of end items for the manufacturing of which the effectivity applies.
- The *effectivity_lot_size* attribute specifies the size of the batch of end items for the manufacturing of which the effectivity applies.

ENTITY lot_effectivity	Attribute Population	Remarks
id	see supertype	
name	see supertype	DERIVEd attribute
description	see supertype	DERIVEd attribute
effectivity_lot_id	type: identifier = string	
effectivity_lot_size	type: entity = measure_with_unit	reference to the measure with unit specifying the size of the lot to be manufactured

Preprocessor Recommendations: In the context of the PDM Schema, lot_effectivity is only used to specify the valid use of a part occurrence in the context of a unit of manufacture of the identified product configuration, i.e. configuration_item. Therefore, it is recommended to only instantiate a lot_effectivity in conjunction, i.e. as complex instance with configuration_effectivity.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

14.2.1.6   serial_numbered_effectivity

A serial_numbered_effectivity is a type of effectivity for which the domain of applicability is defined as a possibly open-ended interval of serial numbers.

Attributes
- The *effectivity_start_id* attribute identifies the first valid serial number for which the effectivity applies.
- The *effectivity_end_id* attribute the last valid serial number for which the effectivity applies. If a value for this attribute is not defined, the interval of applicability has no serial number defined as upper bound, i.e.the effectivity is expected to apply to all serial numbers greater than the effectivity_start_id.

ENTITY serial_numbered_effectivity	Attribute Population	Remarks
id	see supertype	
name	see supertype	DERIVEd attribute
description	see supertype	DERIVEd attribute
effectivity_start_id	type: identifier = string	
effectivity_end_id	type: identifier = string	OPTIONAL

Preprocessor Recommendations: In the context of the PDM Schema, serial_numbered_effectivity is only used to specify the valid use of a part occurrence in the context of a serial number range for end items of the identified product configuration, i.e. configuration_item. Therefore, it is recommended to only instantiate a serial_numbered_effectivity in conjunction, i.e. as complex instance with configuration_effectivity.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#100=PRODUCT_CONCEPT_CONTEXT('pcc_name1', #1, '');
#1=APPLICATION_CONTEXT('');
#2=PRODUCT_CONCEPT('PC-M01', 'PC model name1', 'PC system', #100);
#5=CONFIGURATION_ITEM('PC-Conf1', 'Base Config Europe', 'PC system
standard configuration for Europe', #2, $);
#6=CONFIGURATION_ITEM('PC-Conf2', 'Base Config US', 'PC system standard
configuration for US', #2, $);
#8=APPLICATION_CONTEXT('');
#9=PRODUCT_CONTEXT('', #8, '');
#10=PRODUCT('PC-0023', 'PC system', $, (#9));
#11=PRODUCT_DEFINITION_FORMATION('D', 'description of PC-0023,D', #10);
#12=PRODUCT_DEFINITION_CONTEXT('part definition', #8, 'design');
#14=PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#10, #18, #20, #21,
#22, #23));
#15=CONFIGURATION_DESIGN(#5, #11);
#16=CONFIGURATION_DESIGN(#6, #11);
#18=PRODUCT('MB-0013', 'Mainboard', $, (#9));
#19=PRODUCT_DEFINITION_FORMATION('F', 'description of MB-0013,F', #18);
#20=PRODUCT('PSU-0009', 'Power supply unit', 'Power supply unit 220V',
(#9));
#21=PRODUCT('PSU-0011', 'Power supply unit', 'Power supply unit 110V',
(#9));
#22=PRODUCT('PR-0133', 'CPU', 'Pentium II 233', (#9));
#23=PRODUCT('PR-0146', 'CPU', 'Pentium II 266', (#9));
#24=PRODUCT_DEFINITION_FORMATION('A', 'description of PSU-0009,A',
#20);
#25=PRODUCT_DEFINITION_FORMATION('B', 'description of PSU-0009,B',
#20);
#26=PRODUCT_DEFINITION_FORMATION('B', 'description of PSU-0011,B',
#21);
#27=PRODUCT_DEFINITION_FORMATION('A', 'description of PR-0133,A', #22);
#28=PRODUCT_DEFINITION_FORMATION('C', 'description of PR-0146,C', #23);
#29=PRODUCT_DEFINITION('pc_v1', 'design view on PC-0023,D', #11, #12);
#30=PRODUCT_DEFINITION('mb_v1', 'design view on MB-0013,F', #19, #12);
#31=PRODUCT_DEFINITION('psu1A_v1', 'design view on PSU-0009,A', #24,
#12);
```

```
#33=PRODUCT_DEFINITION('psu1B_v1', 'design view on PSU-0009,B', #25,
#12);
#34=PRODUCT_DEFINITION('psua_v1', 'design view on PSU-0011,B', #26,
#12);
#35=PRODUCT_DEFINITION('pr1_v1', 'design view on PR-0133,A', #27, #12);
#36=PRODUCT_DEFINITION('pr2_v1', 'design view on PR-0146,C', #28, #12);
#37=NEXT_ASSEMBLY_USAGE_OCCURRENCE('mb-u1', 'single instance usage', $,
#29, #30, $);
#38=NEXT_ASSEMBLY_USAGE_OCCURRENCE('psu-u1', 'single instance usage',
$, #29, #31, $);
#39=NEXT_ASSEMBLY_USAGE_OCCURRENCE('psu-u2', 'single instance usage',
$, #29, #33, $);
#40=NEXT_ASSEMBLY_USAGE_OCCURRENCE('psu-u3', 'single instance usage',
$, #29, #34, $);
#41=NEXT_ASSEMBLY_USAGE_OCCURRENCE('cpu-u1', 'single instance usage',
$, #30, #35, $);
#42=NEXT_ASSEMBLY_USAGE_OCCURRENCE('cpu-u2', 'single instance usage',
$, #30, #36, $);
#43=(CONFIGURATION_EFFECTIVITY(#15)
     DATED_EFFECTIVITY(#970, #910)
     EFFECTIVITY('')
     PRODUCT_DEFINITION_EFFECTIVITY(#38)
    );
#45=(CONFIGURATION_EFFECTIVITY(#15)
     DATED_EFFECTIVITY($, #1010)
     EFFECTIVITY('')
     PRODUCT_DEFINITION_EFFECTIVITY(#39)
    );
#46=(CONFIGURATION_EFFECTIVITY(#16)
     EFFECTIVITY('')
     PRODUCT_DEFINITION_EFFECTIVITY(#40)
     SERIAL_NUMBERED_EFFECTIVITY('PS253-000567', $)
    );
#47=(CONFIGURATION_EFFECTIVITY(#15)
     DATED_EFFECTIVITY($, #1160)
     EFFECTIVITY('')
     PRODUCT_DEFINITION_EFFECTIVITY(#41)
    );
#48=(CONFIGURATION_EFFECTIVITY(#16)
     EFFECTIVITY('')
     PRODUCT_DEFINITION_EFFECTIVITY(#41)
     SERIAL_NUMBERED_EFFECTIVITY('PS253-000345', 'PS253-000976')
    );
#49=(CONFIGURATION_EFFECTIVITY(#16)
     EFFECTIVITY('')
     PRODUCT_DEFINITION_EFFECTIVITY(#42)
     SERIAL_NUMBERED_EFFECTIVITY('PS253-000977', $)
    );
#910=DATE_AND_TIME(#920, #930);
#920=CALENDAR_DATE(1998, 1, 7);
#930=LOCAL_TIME(0, 0, 0., #940);
#940=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#970=DATE_AND_TIME(#980, #990);
#980=CALENDAR_DATE(1999, 31, 3);
#990=LOCAL_TIME(0, 0, 0., #1000);
#1000=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#1010=DATE_AND_TIME(#1020, #1030);
```

```
#1020=CALENDAR_DATE(1999, 1, 4);
#1030=LOCAL_TIME(0, 0, 0., #1031);
#1031=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#1110=DATE_AND_TIME(#1120, #1130);
#1120=CALENDAR_DATE(1998, 15, 7);
#1130=LOCAL_TIME(0, 0, 0., #1140);
#1140=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#1160=DATE_AND_TIME(#1170, #1180);
#1170=CALENDAR_DATE(1998, 1, 10);
#1180=LOCAL_TIME(0, 0, 0., #1190);
#1190=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
```

Example 64 : exchange file segment for configuration effectivity

14.3 General Validity Period

The PDM Schema allows assignment of general validity periods to product data that control the usage of these product data independent of any particular usage. The object to which the general validity period effectivity is assigned is considered valid within the identified period of time for any usage and within all contexts.

Configuration effectivity is used to define the planned usage of components in the context of a particular product configuration. If both general validity period and configuration effectivity is assigned to a particular component, the more restrictive of the assigned effectivities should be considered as in effect.

14.3.1 General validity period effectivity

The general validity period effectivity is used to describe effectivities with respect to a certain time period in which the associated product data may be used. The object to which the general validity period effectivity is assigned is considered valid within the identified period of time.

The general validity effectivity may be applied in two ways:
- usage independent general validity period,
- usage dependent general validity period.

Usage independent general validity period effectivity can be applied to 'stand-alone' elements independent from their usage by some other element, such as part (or document) versions or particular views on these. If a general validity period effectivity is assigned to a product_definition_formation, it is assumed that this effectivity also applies to all associated product_definitions that do not carry their own effectivity information.

Usage dependent general validity period effectivity can be applied to product_definition_relationship instances representing a usage of the related_product_definition in the context of the relating_product_definition. It should be used to define the validity period in which the related_product_definition may be used/is to be used in the context of the relating_product_definition.

Usage dependent and usage independent effectivity concepts are referred to as structure related and version related effectivity. Either one can be used to describe a valid product data structure for a certain time range The two concepts can also be used in combination, but restrictions should apply for the defined validity periods. If different effectivities apply to a product_definition and a product_definition_relationship where the product_definition is the related_product_definition, then the effectivity period of the product_definition_relationship should be within the effectivity period defined for the related_product_definition.

The general validity period effectivity is based on the definition of the effectivity entity as defined in 14.2.1.1. Two types of effectivity may be used to describe general validity period effectivity:

- dated_effectivity, i.e. the validity period can be defined by the points in time when the period of applicability of the associated object starts and ends (see 14.2.1.4),
- time_interval_based_effectivity, i.e.the validity period is defined as an intervening time.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements of general validity period effectivity are illustrated in Diagram 66 below. The corresponding STEP exchange file encoding is given in Example 65 and Example 66.
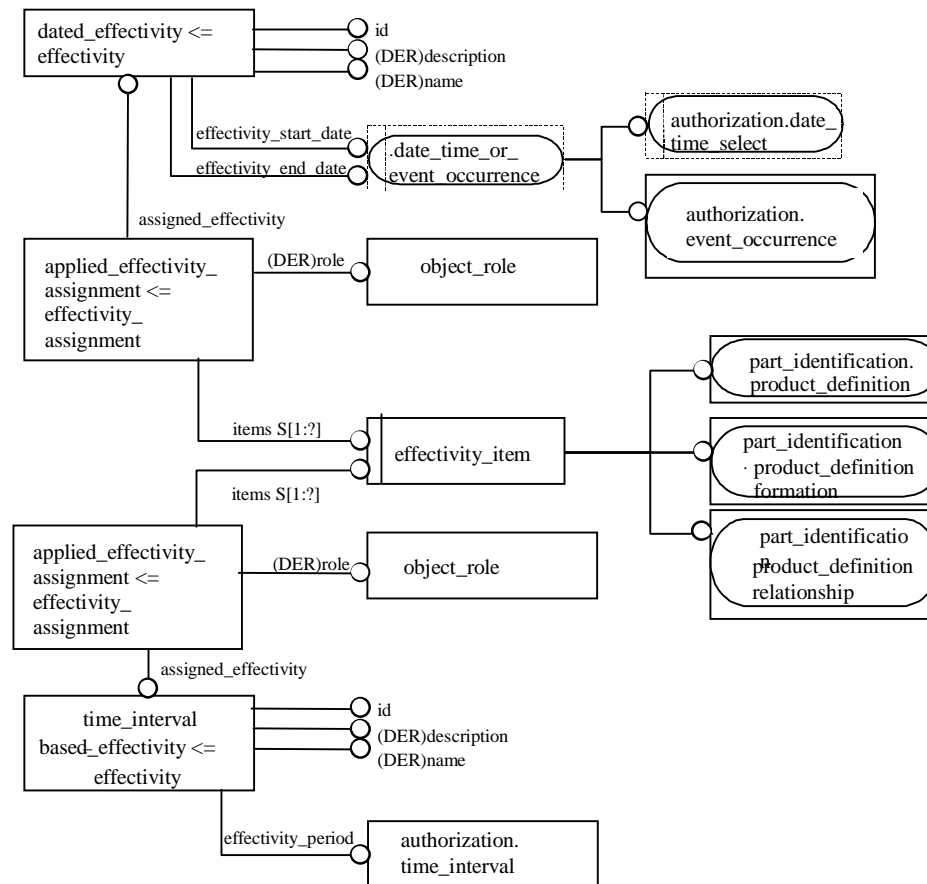
Diagram 66: General validity period effectivity Instance Diagram

14.3.1.1   time_interval_based_effectivity

The entity time_interval_based_effectivity is a subtype of effectivity for which the domain of applicability is defined as a possibly open **time_interval**. This type of effectivity enables the specification of an interval of time through one boundary and the duration of the interval.

Attributes
- The *effectivity_period* attribute identifies the time_interval in which the associated product data is effective.

ENTITY time_interval_based_effectivity	Attribute Population	Remarks
id	see supertype	
name	see supertype	DERIVEd attribute
description	see supertype	DERIVEd attribute
effectivity_period	type: entity = time_interval	

Preprocessor Recommendations: In the context of the PDM Schema, time_interval_based_effectivity is only used to general validity periods for the associated product data. Therefore, it should not be instantiated in conjunction, i.e. as complex instance with configuration_effectivity.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

14.3.1.2   effectivity_relationship

An Effectivity relationship is a relationship between two effectivity objects. The meaning of the effectivity relationship is given by its *name* attribute. An effectivity_relationship may be used to define an effectivity with respect to some other effectivity or to express some dependency between two effectivity objects.

Attributes
- The *name* attribute specifies the meaning of the relationship.
- The *description* attribute specifies additional information text that characterizes the effectivity_relationship.
- The *relating_effectivity* specifies the first effectivity that is related by the effectivity_relationship. The relating_effectivity usually identifies the effectivity the definition of the related_effectivity is based on, e.g. derived from or dependent on. The semantics of this attribute should be defined by the name attribute.
- The *related_effectivity* specifies the second effectivity related by the effectivity_relationship. The related_effectivity usually identifies the effectivity which is based on the definition of the relating_effectivity. The semantics of this attribute should be defined by the name attribute.

ENTITY effectivity_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	OPTIONAL
relating_effectivity	type: entity = effectivity	
related_effectivity	type: entity = effectivity	

PreprocessorRecommendations: The name attribute is used to specify the type of the effectivity_relationship and therefore should be populated with predefined values that both sides of the exchange have a common understanding of. The value 'constraint' for the name attribute should be used to indicate that the validity period defined by the related_effectivity should be within the time period of the relating_effectivity. The value 'inheritance' for the name attribute should be used to indicate that the related_effectivity shall not define its own validity period, but inherits the validity period from the relating_effectivity (this inheritance might occur over a chain of interrelated effectivity_relationship instances). There are no further standard mappings specified for the name attribute. Additional values must be mutually agreed upon.

There is no standard mapping for the description attribute. The description attribute is optional, but may contain any appropriate/mutually agreed upon string. If this attribute is not needed/used, it should be assigned the value '$'.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

14.3.1.3   applied_effectivity_assignment

The general validity period effectivity may be applied to different types of product data, potentially specifying an additional role in which the effectivity is assigned. The effectivity_assignment entity defines a general mechanism to associate general validity period to some product data.

The entity applied_effectivity_assignment is a subtype of effectivity assignment.   It adds the attribute items, which refers to the instances of product data to which the validity period is assigned. The types of product data to which general validity period effectivity may be assigned are defined in the effectivity_item select type.

The applied_effectivity_assignment entity should have a role associated with it through the entity object_role. If a role is associated, the applied_effectivity_assignment is referenced as the item_with_role by a role_association where the role is stored in the name attribute of the object_role associated as role. Role information may be used to distinguish between e.g. required, planned, or actual effectivities of an object.

Attributes
- The *assigned_effectivity* specifies the effectivity, which is assigned.
- The *items* attribute specifies the set of items to which the associated effectivity is assigned.

ENTITY applied_effectivity_assignment	Attribute Population	Remarks
assigned_effectivity	type: entity = effectivity	
role	type: entity = object_role	DERIVEd attribute
items	type: entity = effectivity_item select	SET[1:?]

Preprocessor Recommendations: applied_effectivity_assignment shall only be used to assign general validity period effectivities, i.e. the assigned effectivity shall not be of type lot_effectivity or serial_numbered_effectivity.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An applied_effectivity_assignment may be referenced by the following entities:
- role_association to specify a object_role defining the role of the assignment.

Where applicable the following role values shall be used:
- 'actual period': The actual period during which the Effectivity lasted;
- 'actual start': The actual date in the past when the Effectivity became effective;
- 'actual stop': The actual date in the past when the Effectivity ceased to be effective;
- 'planned period': The period associated with the Effectivity defines a planned period of time during which the associated object is or was supposed to be effective;
- 'planned start': The date or event associated with the Effectivity defines the date or event when the Effectivity is or was supposed to start;
- 'planned stop': The date or event associated with the Effectivity defines the date or event when the Effectivity is or was supposed to stop;
- 'required period': The associated object must be kept effective for this period;
- 'required start': The date or event associated with the Effectivity is a due start date. Conformance to this bound is expected;
- 'required stop': The date or event associated with the Effectivity is a due stop date. Conformance to this bound is expected.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#1=APPLICATION_CONTEXT('mechanical design');
#220=PRODUCT_CONTEXT('', #1, '');
#230=PRODUCT_DEFINITION_CONTEXT('part definition', #1, 'design');
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#2));
/* part master definition with two succeeding versions*/
#2=PRODUCT('11000', 'solid cube', 'description for part 11000',
(#220));
#3=PRODUCT_DEFINITION_FORMATION('A', 'version A for solid cube', #2);
#4=PRODUCT_DEFINITION('D1', 'detailed drawing as planned for STEP
conformance testing', #3, #230);
#10=PRODUCT_DEFINITION_FORMATION('B', 'version B for part 11000', #2);
#11=PRODUCT_DEFINITION('D2', 'detailed drawing', #10, #230);
#16=PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('id', 'sequence', $, #3,
#10);
/* start and end dates for effectivity */
#910=DATE_AND_TIME(#920, #930);
#920=CALENDAR_DATE(1998, 1, 7);
#930=LOCAL_TIME(0, 0, 0., #940);
#940=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#970=DATE_AND_TIME(#980, #990);
#980=CALENDAR_DATE(1999, 31, 3);
#990=LOCAL_TIME(0, 0, 0., #1000);
#1000=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#1010=DATE_AND_TIME(#1020, #1030);
#1020=CALENDAR_DATE(1999, 1, 4);
#1030=LOCAL_TIME(0, 0, 0., #1031);
#1031=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
/* effectivity definition and assignment */
#1040=DATED_EFFECTIVITY('', #970, #910);
#1041=DATED_EFFECTIVITY('', $, #1010);
#1042=APPLIED_EFFECTIVITY_ASSIGNMENT(#1040, (#3));
#1043=APPLIED_EFFECTIVITY_ASSIGNMENT(#1041, (#10));
#1044=OBJECT_ROLE('actual', $);
#1045=ROLE_ASSOCIATION(#1044, #1042);
#1046=ROLE_ASSOCIATION(#1044, #1043);
```

Example 65: exchange file segment for view independent general validity period effectivity

```
#10=APPLICATION_CONTEXT('mechanical design');
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #440));
#220=PRODUCT_CONTEXT('', #10, '');
#230=PRODUCT_DEFINITION_CONTEXT('part definition', #10, 'design');
#340=PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350=PRODUCT_DEFINITION_CONTEXT('assembly definition', #10, '');
#360=PRODUCT_RELATED_PRODUCT_CATEGORY('detail', $, (#380));
#380=PRODUCT('g1', 'gasket', $, (#220));
#390=PRODUCT_DEFINITION_FORMATION('A', '', #380);
#400=PRODUCT_DEFINITION('gv1', 'design view on gasket A', #390, #230);
#420=PRODUCT_RELATED_PRODUCT_CATEGORY('assembly', $, (#440));
#440=PRODUCT('s1', 'sleeve assembly', $, (#220));
#450=PRODUCT_DEFINITION_FORMATION('A', '', #440);
#460=PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly', #450,
#230);
```

```
#530=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu-1', 'single instance usage',
'gasket usage left', #460, #400, $);
#540=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu-2 ', 'single instance usage',
'gasket usage right', #460, #400, $);
#800=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
#810=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#815, #350, #340);
#811=PRODUCT_DEFINITION_FORMATION('B', 'gasket modified', #380);
#812=PRODUCT_DEFINITION_FORMATION_RELATIONSHIP('', 'sequence', $, #390,
#811);
#813=PRODUCT_DEFINITION_CONTEXT('part definition', #10,
'manufacturing');
#814=PRODUCT_DEFINITION('gv3', 'manufacturing view on gasket A', #390,
#813);
#815=PRODUCT_DEFINITION('sv2', 'manufacturing view on sleeve assembly',
#450, #813);
#816=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu3', 'single instance usage',
'gasket usage left', #815, #814, $);
#817=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu4', 'single instance usage',
'gasket usage right', #815, #814, $);
#818=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu5', 'single instance usage',
'gasket usage left', #460, #821, $);
#819=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu6', 'single instance usage',
'gasket usage right', #460, #821, $);
#821=PRODUCT_DEFINITION('gv2', 'design view on gasket B', #811, #230);
#910=DATE_AND_TIME(#920, #930);
#920=CALENDAR_DATE(1998, 1, 7);
#930=LOCAL_TIME(0, 0, 0., #940);
#940=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#970=DATE_AND_TIME(#980, #990);
#980=CALENDAR_DATE(1999, 31, 3);
#990=LOCAL_TIME(0, 0, 0., #1000);
#1000=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#1010=DATE_AND_TIME(#1020, #1030);
#1020=CALENDAR_DATE(1999, 1, 4);
#1030=LOCAL_TIME(0, 0, 0., #1031);
#1031=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#1040=DATED_EFFECTIVITY('', #970, #910);
#1041=DATED_EFFECTIVITY('', $, #1010);
#1042=APPLIED_EFFECTIVITY_ASSIGNMENT(#1040, (#400));
#1043=APPLIED_EFFECTIVITY_ASSIGNMENT(#1041, (#821));
#1044=OBJECT_ROLE('actual', $);
#1045=ROLE_ASSOCIATION(#1044, #1042);
#1046=ROLE_ASSOCIATION(#1044, #1043);
#1050=DATED_EFFECTIVITY('', $, #1110);
#1051=DATED_EFFECTIVITY('', $, #1160);
#1052=APPLIED_EFFECTIVITY_ASSIGNMENT(#1050, (#460));
#1053=APPLIED_EFFECTIVITY_ASSIGNMENT(#1051, (#815, #814));
#1110=DATE_AND_TIME(#1120, #1130);
#1120=CALENDAR_DATE(1998, 15, 7);
#1130=LOCAL_TIME(0, 0, 0., #1140);
#1140=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
#1160=DATE_AND_TIME(#1170, #1180);
#1170=CALENDAR_DATE(1998, 1, 10);
#1180=LOCAL_TIME(0, 0, 0., #1190);
#1190=COORDINATED_UNIVERSAL_TIME_OFFSET(1, $, .AHEAD.);
```

Example 66: exchange file segment for general validity period effectivity applied to views

15 Engineering Change and Work Management

The PDM Schema provides data structures for representation of the data used to manage the work being done during the engineering (release and change) process. The work management area contains the constructs to describe initial part design requirements and the change requirements and issues for revising part designs, as well as the directive for work and actions to proceed in the development of these initial or modified part designs. The structures are based on three fundamental concepts: work request, work order and activity.

A work request documents the need for a potential action such as creation of new functionality (release) or change of existing functionality, which may or may not ever be incorporated. If the request is incorporated, it is done through some action being taken on the request, which may result in the release of new functionality or a change to some existing functionality (or of portions thereof). This action identifying the work to be done to address the request is formally directed by a work order.

All release and change proposals such as Engineering Change Proposals, Requests for Engineering Action, etc., are represented by the request for work portion of the work management data structure, i.e. the versioned_action_request and its related entity types. For a release or change proposal to be incorporated, there must be a definition of the actual work to be done. The work definition area includes the definition of work orders together with the activities that are directed by the work order.

In addition to the request for work and work definition structures, the PDM Schema provides the capability to represent project and contract related information, and to associate the work being done as part of the engineering process to the projects and contracts that control it.

15.1 Request for Work

The work done as part of an engineering process may be triggered or initiated by official requests that identify the need for a potential release or change of functionality. These requests may or may not ever actually be incorporated, depending upon the level of authorization granted to the request and its associated work order.

The STEP PDM Schema supports the representation of these work requests including their level of completion, the reason for the suggested work to be done, and the desired outcome. The items that are planned as the possible subject of this work may be identified as in scope and attached to the work request. If the request is in fact incorporated, these planned affected items of the work request may or may not actually be the items affected by the associated work order. One ore more potential alternative solutions for addressing the work request may also be specified.

Additional administrative information such as persons, organizations, and date and time related information as well as approval authorization information can also be attached to a work request.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements to represent request for work information are illustrated in Diagram 67 below. The corresponding STEP exchange file encoding is given in Example 67.
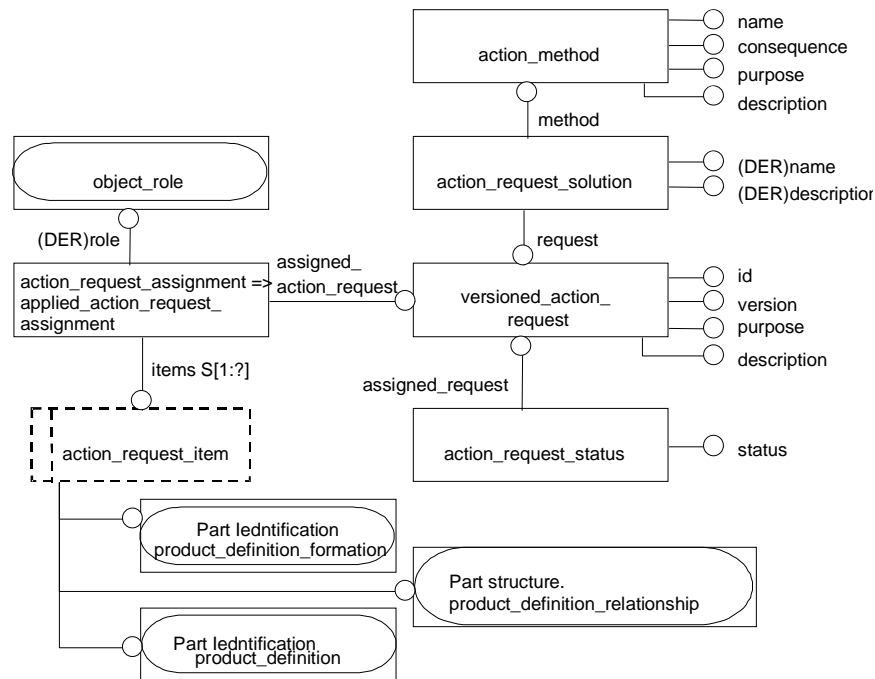
Diagram 67 : Request for Work Instance Diagram

15.1.1.1   versioned_action_request

The versioned_action_request entity represents a request for some work which may be an initial release or a change. A versioned_action_request also specifies the reason for this work.

Possible ways to achieve the purpose of the request, i.e. potential alternative solutions, may be identified as one or more action_method instances that are attached to the versioned_action_request using the entity action_request_solution (see 15.1.1.4). Status information describing the level of completion of the versioned_action_request must be represented through an associated action_request_status (see 15.1.1.2).

The items that are planned as the subject of the work request are associated to the work request using applied_action_request_assignment (see 15.1.1.3). For example, if a versioned_action_request represents a request for change the items to be changed are identified through an applied_action_request_assignment. Such items may be part or document versions or view definitions.

Attributes
- The *id* attribute specifies an identifier that distinguishes the versioned_action_request.
- The *version* attribute specifies the identification of the version of the versioned_action_request.
- The *purpose* attribute specifies a descriptive label providing an informal description of the reason for the versioned_action_request.
- The *description* attribute specifies additional information **about** the versioned_action_request.

ENTITY versioned_action_request	Attribute Population	Remarks
id	type: identifier = string	must be unique in conjunction with the version attribute
version	type: label = text	
purpose	type: text = string	
description	type: text = string	OPTIONAL

Preprocessor Recommendations: There is no standard mapping for the id attribute of versioned_action_request; however the value should be unique in conjunction with the version attribute.

The purpose attribute is mandatory, and may contain any appropriate/mutually agreed upon string. Where applicable, the following values for the purpose attribute may be used: 'technical improvement', 'government regulation', 'durability improvement', 'tool improvement', 'production relief', 'production requirement', 'quality improvement', 'security reason', 'standardization', 'cost reduction', 'customer rejection', 'change of standard', 'production alignment', and 'procurement alignment'.

A versioned_action_request shall have an associated status, represented by exactly one action_request_status, which references the versioned_action_request as the assigned_request (see 15.1.1.2).

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A versioned_action_request may be referenced by the following entities:
- action_request_solution to identify potential solutions to address the versioned_action_request.
- applied_action_request_assignment to identify the items that are planned as the subject of the assigned work request.
  - Note: If a versioned_action_request is referenced as the assigned_request by an applied_action_request_assignment specifying one or more product_definition_formations, product_definitions, or product_definition_relationships as the affected items, it is assumed that the versioned_action_request represents a request for change of the referenced objects, otherwise it represents a request for an initial design.
- action_directive to identify the definition of related engineering action(s) that provide the work definition proposed to incorporate the purpose of the versioned_action_request.
- applied_approval_assignment to associate approval information related to a versioned_action_request. See 13.2 for guidance in creating the approval constructs.
- applied_date_assignment (or applied_date_and_time_assignment) to associate date information to a versioned_action_request. See 13.3.1 for guidance in creating the date and time constructs. A recommended value for the name attribute of the date_role is 'request date'.
- applied_organization_assignment (or applied_person_and_organization_assignment) to associate organization (and person) information to a versioned_action_request. See 13.1 for guidance in creating the organization and person constructs. A recommended value for the name attribute of the organization_role (or person_and_organization_role) is 'requestor'.

15.1.1.2   action_request_status

The action_request_status entity represents the status describing the level of completion of the associated **versioned_action_request**.

Attributes
- The *status* attribute specifies a label that provides a user interpretable designation for the level of completion of the versioned_action_request.
- The *assigned_request* attribute specifies the versioned_action_request for which the status is specified.

ENTITY action_request_status	Attribute Population	Remarks
status	type: label = text	
assigned_request	type: entity = versioned_action_request	

Preprocessor Recommendations: The *status* attribute characterizes the level of completion of the versioned_action_request and therefore should be populated with predefined values that are commonly understood by both sides of the exchange. Recommended values to be used are "proposed", "in-work", "issued", and "hold".

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

15.1.1.3   applied_action_request_assignment

A work request may be raised against different types of product data that might be subject to the requested work. The applied_action_request_assignment entity defines a mechanism to associate a work request with the items that are thought or planned to be affected by the work request.

The STEP PDM Schema allows versions, views, and structure relationships of parts and documents (i.e. product_definition_formation, product_definition, and product_definition_relationship instances) to be subject of a work request.

Attributes
- The *assigned_action_request* specifies the versioned_action_request, which is assigned to the affected product data.
- The *items* attribute specifies the set of items that are subject of the work request represented by the associated versioned_action_request.

ENTITY applied_ action_request_assignment	Attribute Population	Remarks
assigned_action_request	type: entity = versioned_action_request	
role	DERIVEd attribute type: entity = object_role	OPTIONAL
items	type: entity = action_request_item select	SET[1:?]

Preprocessor Recommendations: The applied_action_request_assignment entity may have a role associated with it through the entity object_role. Where applicable, the value 'scope' should be used **to indicate that the** referenced items are planned in scope of the assigned work request. If there is no role specified for an applied_action_request_assignment, it is assumed that the versioned_action_request represents a change request for the referenced objects.

Postprocessor Recommendations: If no object_role is specified for the applied_action_request_- assignment, the assigned versioned_action_request shall be interpreted as a change request requesting a change of the associated items ('scope').

Related Entities: An action_request_assignment may be referenced by the following entities:
- role_association to specify an object_role defining the role of the assignment, i.e. purpose of the association of the applied_action_request_assignment **with product data**.

15.1.1.4   action_request_solution

The action_request_solution entity represents an association between a work and an action_method that describes a potential solution for the request. Identified solutions of a versioned_action_request may be explicitly recommended or explicity excluded from a recommendation.

Attributes
- The *method* attribute specifies **an** action_method that is a potential solution for the request.
- The *request* attribute specifies **the versioned_action_request for which the (potential) solution is specified**.

ENTITY action_request_solution	Attribute Population	Remarks
method	type: entity = action_method	
request	type: entity = versioned_action_request	
description	DERIVEd attribute type: text = string	at most one description can be associated via the entity description_attribute
name	DERIVEd attribute type: label = string	at most one name can be associated via the entity name_attribute

Preprocessor Recommendations: If a potential solution identified by an action_request_solution is explicitly recommended to address the issue raised by the versioned_action_request, the value 'recommended method' may be used for the name attribute. If a potential solution is identified as such, but not recommended to solve the work request, the value 'non recommended method' may be used for the name attribute.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

15.1.1.5   action_method

The action_method entity defines relevant information about an activity. This definition includes the activity's objectives and effects and describes how the activity is to be carried out. Within the work management area, action_methods may be used to describe procedures for how to carry out activities and to represent the potential solutions to address a work request.

Attributes
- The *name* attribute specifies the word or common name used to refer to the action_method.
- The *description* attribute specifies **additional information about** the action_method.
- The *consequence* attribute contains an informal description of the effects of the action_method.
- The *purpose* attribute specifies a descriptive label characterizing the objectives of the action_method.

ENTITY action_method	Attribute Population	Remarks
name	type: label = text	
description	type: text = string	OPTIONAL
consequence	type: text = string	
purpose	type: label = text	

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An action_method may be referenced by the following entities:
- executed_action and directed_action to indicate the activities for which the action_method describes the procedure.
- action_request_solution to specify the versioned_action_request(s) for which the action_method defines a potential solution.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#1=VERSIONED_ACTION_REQUEST('var id1', '0001', 'create new part
design', '$');
#2=VERSIONED_ACTION_REQUEST('var id2', '0001', 'technical improvement',
'description of var id2, 0001');
#3=VERSIONED_ACTION_REQUEST('var id3', '0002', 'production
requirement', 'description of var id3, 0002');
#4=ACTION_REQUEST_STATUS('proposed', #1);
#5=ACTION_REQUEST_STATUS('issued', #2);
#6=ACTION_REQUEST_STATUS('in-work', #3);
#8=ACTION_METHOD('activity method 1', $, 'effects of activity method
1', 'objectives of activity method 1');
#9=ACTION_METHOD('technical improvement method 1', $, 'effects of
technical improvement method 1', '$');
#10=ACTION_METHOD('technical improvement method 2', $, 'effects of
technical improvement method 2', '$');
#13=ACTION_REQUEST_SOLUTION(#8, #1);
#14=ACTION_REQUEST_SOLUTION(#9, #2);
#15=ACTION_REQUEST_SOLUTION(#10, #2);
#16=APPLICATION_CONTEXT('mechanical design');
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part', $, (#380, #440));
#220=PRODUCT_CONTEXT('', #16, '');
#230=PRODUCT_DEFINITION_CONTEXT('part definition', #16, 'design');
#340=PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350=PRODUCT_DEFINITION_CONTEXT('assembly definition', #16, '');
#380=PRODUCT('g1', 'gasket', $, (#220));
#390=PRODUCT_DEFINITION_FORMATION('A', '', #380);
#400=PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);
#440=PRODUCT('s1', 'sleeve assembly', $, (#220));
#450=PRODUCT_DEFINITION_FORMATION('A', '', #440);
#460=PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly', #450,
#230);
#520=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
#530=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu1', 'single instance usage',
'gasket usage 1', #460, #400, $);
#531=APPLIED_ACTION_REQUEST_ASSIGNMENT(#2, (#450));
#532=APPLIED_ACTION_REQUEST_ASSIGNMENT(#3, (#400));
```

Example 67 : exchange file segment for request for work

15.2 Work Order and Work Definition

Requests for work are typically followed by a definition of the engineering activities required to address the issues raised by the work requests. The engineering activity that defines a work order is formally directed in response to a work request.

Every work order consists of a work directive (in response to a work request) and a directed activity that defines the work to be done. This work definition represents the main activity that is directed by and has to be carried out as part of a work order.

The main activity describing the work definition related to a work order may be further refined by decomposition into sub-activities. This is described in the section on activity decomposition (see 15.2.2).

15.2.1 Work Order

Requests for work are processed and result in work orders that direct engineering activities to solve the issues raised by the work request. A work order may address one or multiple work requests.

The work order describes how a particular request for work is addressed and, if authorized, incorporated through appropriate activities during the engineering release or change process. Every work order consists of a work directive (directed in response to a specific work request) and a definition of the work to be done. This work definition represents the main activity defining the work to be completed under a work order, together with a reference to the activity method that identifies the procedure describing how it is to be carried out.

An activity may be associated with status information describing its level of completion. The identification of the items that are the input or the result of an engineering activity that is controlled by a work order is also supported by the STEP PDM Schema.

In addition, person, organization and date and time related information as well as approval information can be attached to an activity.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements to represent work order information are illustrated in Diagram 68 below. The corresponding STEP exchange file encoding is given in Example 68.
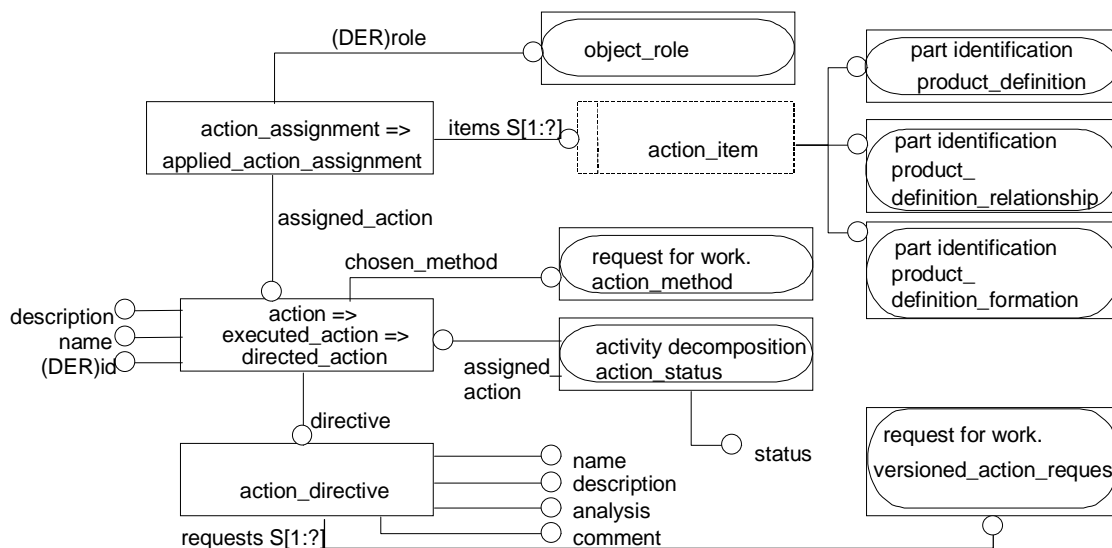

Diagram 68: Work order Instance Diagram

15.2.1.1   action_directive

The action_directive entity specifies an authoritative instrument that provides directions to achieve the desired results as specified in some work_request(s). An action_directive is the result of processing one or more versioned_action_requests. It represents the directive for work that must be performed to satisfy the requirements stated by the assessed versioned_action_requests.   The work directed by an action_directive is represented by an associated directed_action (see 15.2.1.2). The action_directive together with the directed activity defining the work to be completed represents a work order.

Attributes
- The *name* attribute specifies the identifier used to refer to the action_directive.
- The *description* attribute specifies **additional information about** the action_directive.
- The *analysis* attribute specifies an informal description of the results of the analysis that was carried out on the elements of the versioned_action_requests that are addressed by the action_directive.
- The *comment* attribute specifies a descriptive label providing some textual commentary.
- The *requests* attribute specifies the work requests that are addressed by the action_directive.

ENTITY action_directive	Attribute Population	Remarks
name	type: label = text	
description	type: text = string	OPTIONAL
analysis	type: text = string	
comment	type: label = text	
requests	type: entity = versioned_action_request	S[1:?]

Preprocessor Recommendations: In the context of the work management area, an action_directive shall be referenced by exactly one directed_action that defines the work that is directed by the action_directive.

The name attribute shall contain a string that, in conjunction with the associated versioned_action_requests, uniquely identifies the action_directive. If an action_directive addresses one particular versioned_action_request, the value of the id attribute of the versioned_action_request may be used to populate the name attribute.

Some types of (release or change) processes in organizations may not involve submission of official work requests. In this case, it is recommended to use the default values for the entities and attributes not supported by the preprocessor as described in the 'General information' section of this usage guide to instantiate the required versioned_action_request.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An action_directive may be referenced by the following entities:
- directed_action to specify the work that is being directed by the action_directive, and that addresses the requirements specified in the associated versioned_action_requests. It is strongly recommended that an action_directive has exactly one associated directed_action with name 'work definition'.

15.2.1.2   directed_action

The directed_action entity represents the work that is controlled by an action_directive, and defined as part of a work order. This is a definition of work that has either taken place, is taking place, or is expected to take place in the future. The work requests that are addressed and incorporated by the directed_action are specified through association to an action_directive.

A directed_action identifies the action_method that describes the method or procedure chosen to carry out the activity. Reference to an action_method that has been identified as a potential solution to a work request (see 15.1.1.4) indicates which procedure or potential solution for a work request has been selected by the work order.

A directed_action may have an associated status representing the level of completion of the work. In addition, it may carry additional information needed to characterize the work defined by the work order, e.g. start or end dates, person and organization related information, and approval authorization information.

A directed_action is related via an applied_action_assignment (see 15.2.1.3) to the items that are subject of the work represented by the 'work definition' of a work order. This is used to describe the items (e.g. part

or document versions, views, or structural relationships between those can), which are either input or result from the release or change defined by the directed_action.

It is recommended that exactly one directed_action be used to represent the work being governed by an action_directive. This directed_action may be identified by the name 'work definition', and may be further decomposed into sub-activities represented by executed_actions (see 15.2.2). These sub-activities are related to the work definition (directed_action) through an action_relationship (or a chain of such action_relationships) with name 'decomposition' (see 15.2.2.3). Each sub-activity may have additional information, e.g. start or end dates, person and organization related information, or approvals specifically assigned to it. When many requests are incorporated by one work order using different proposed solutions, the directed_action representing the complete work definition should be decomposed in sub-activities for which the appropriate method is indicated.

Attributes
- The *name* attribute specifies the word or common name used to refer to the action.
- The *description* attribute specifies **additional information about** the action.
- The *chosen_method* attribute specifies the action_method that defines the procedure selected to carry out the **action** and a description of the results of the **action**.
- The *directive* attribute specifies the action_directive that authorizes and controls the directed_action.
- The *id* attribute is derived, and may store the unique identifier for a directed_action.

ENTITY directed_action	Attribute Population	Remarks
name	type: label = text	may be 'work definition'
description	type: text = string	OPTIONAL
chosen_method	type: entity = action_method	
directive	type: entity = action_directive	
id	DERIVEd attribute type: identifier = string	exactly one id may be associated via the entity id_attribute

Preprocessor Recommendations: It is recommended that a directed_action always be related to an action_directive. The name attribute shall be unique in conjunction with the associated action_directive.

If the directed_action representing the 'work definition' is broken down in separate sub-activities (see 15.2.2), it is recommended that this top-level work definition be associated through applied_action_assignment to all input and output items that are defined for the sub-activities.

It is recommended that a directed_action have an associated status, represented by exactly one action_status, which references the directed_action as the assigned_action (see 15.2.2.2).

The directed_action entity may have an identifier associated with it through the entity id_attribute and its attribute_value attribute.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A directed_action may be referenced by the following entities:
- action_relationship to associate two directed_actions or a directed_action and an executed_action e.g. to define their decomposition relationship, i.e. one action as part of the other, or to describe their dependencies.
- applied_action_assignment to identify the items that are affected by the directed_action.
- applied_approval_assignment to associate authorization information related to a directed_action. See 13.2 for guidance in creating the approval constructs.
- applied_date_assignment (or applied_date_and_time_assignment) to associate date (and time) information to a directed_action. See 13.3.1 for guidance in creating the date and time constructs.
- applied_organization_assignment (or applied_person_and_organization_assignment) to associate organization (and person) information to a directed_action. See 13.1 for guidance in creating the

organization and person constructs.   A recommended value for the name attribute of the person_and_organization_role is 'requestor'.

- applied_contract_assignment to identify that the directed_action is subject of the assigned contract.

15.2.1.3   applied_action_assignment

The applied_action_assignment entity relates an action to the elements of product data that is affected by the action. The applied_action_assignment entity defines a general mechanism to associate an activity that is part of a work order to the items that are affected by the activity.

The types of items to which an activity may be assigned are defined in the action_item select type. Within the scope of the PDM Schema, items that may be affected by an engineering activity are part and document versions (i.e. product_definition_formations), views on these versions (i.e. product_definitions), or structural relationships between these (i.e. product_definition_relationships).

The applied_action_assignment may have a role associated with it through the entity object_role via a role_association. The role specifies the function that is performed by the affected item in the context of the assigned action. Role information may be used to distinguish between items that are used as input or the ones that are the results from the release or change process.

Attributes
- The ***assigned_action*** specifies the action that is assigned to the affected items.
- The ***items*** attribute specifies the set of items to which the associated action is assigned, i.e. that are affected by the assigned action.

ENTITY applied_ action _assignment	Attribute Population	Remarks
assigned_action	type: entity = action	
role	DERIVEd attribute type: entity = object_role	OPTIONAL
items	type: entity = action_item select	SET[1:?]

Preprocessor Recommendations: If the applied_action_assignment has a role associated with it through the entity object_role, the name attribute of the object role should use the following values where applicable: 'input' should be used to indicate that the referenced item serves as initial data for the assigned action; 'output' should be used to indicate that the referenced item is a result of the assigned action. If no role is given, it is assumed that the applied_action_assignment identifies items that are the result of carrying out the assigned action (e.g. 'output').

If a directed_action defining the complete work under a work order is further decomposed in sub-activities, then all affected items identified as input or output of the sub-activities should be explicitly associated to the top-level directed_action.

Postprocessor Recommendations: If the applied_action_assignment has no role associated with it, it shall be assumed that the items referenced by the applied_action_assignment are the result of carrying out the assigned action.

Related Entities: An applied_action_assignment may be referenced by the following entities:
- role_association to specify an object_role defining the role of the assignment.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#1=VERSIONED_ACTION_REQUEST('var id1', '0001', 'create new part
design', '$');
```

```
#2=VERSIONED_ACTION_REQUEST('var id2', '0001', 'technical improvement',
'description of var id2, 0001');
#4=ACTION_REQUEST_STATUS('in-work', #1);
#5=ACTION_REQUEST_STATUS('issued', #2);
#8=ACTION_METHOD('activity method 1', $, 'effects of activity method
1', 'objectives of activity method 1');
#9=ACTION_METHOD('technical improvement method 1', $, 'effects of
technical improvement method 1', '$');
#10=ACTION_METHOD('technical improvement method 2', $, 'effects of
technical improvement method 2', '$');
#13=ACTION_REQUEST_SOLUTION(#8, #1);
#14=ACTION_REQUEST_SOLUTION(#9, #2);
#15=ACTION_REQUEST_SOLUTION(#10, #2);
#16=APPLICATION_CONTEXT('mechanical design');
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#380, #440, #542));
#220=PRODUCT_CONTEXT('', #16, '');
#230=PRODUCT_DEFINITION_CONTEXT('part definition', #16, 'design');
#340=PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350=PRODUCT_DEFINITION_CONTEXT('assembly definition', #16, '');
#380=PRODUCT('g1', 'gasket', $, (#220));
#390=PRODUCT_DEFINITION_FORMATION('A', '', #380);
#400=PRODUCT_DEFINITION('gv1', 'design view on gasket', #390, #230);
#440=PRODUCT('s1', 'sleeve assembly', $, (#220));
#450=PRODUCT_DEFINITION_FORMATION('A', '', #440);
#460=PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly', #450,
#230);
#520=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
#530=NEXT_ASSEMBLY_USAGE_OCCURRENCE('gu1', 'single instance usage',
'gasket usage 1', #460, #400, $);
#531=APPLIED_ACTION_REQUEST_ASSIGNMENT(#2, (#450));
#532=APPLIED_ACTION_REQUEST_ASSIGNMENT(#547, (#400));
#547=VERSIONED_ACTION_REQUEST('var id3', '0002', 'production
requirement', 'description of var id3, 0002');
#533=ACTION_DIRECTIVE('wo id1', 'design release', 'results of analysis
of var id1', 'comment 1', (#1));
#534=ACTION_DIRECTIVE('wo id2', 'design change', 'results of analysis
of var id2', 'comment 2', (#2));
#535=DIRECTED_ACTION('design', $, #537, #533);
#536=DIRECTED_ACTION('work definition', $, #10, #534);
#537=ACTION_METHOD('activity method 2', $, 'effects of activity method
2', 'objectives of activity method 2');
#538=PRODUCT_DEFINITION_FORMATION('B', '', #440);
#539=PRODUCT_DEFINITION('new_part_v1', 'design view on new part', #543,
#230);
#540=APPLIED_ACTION_ASSIGNMENT(#535, (#539));
#541=APPLIED_ACTION_ASSIGNMENT(#536, (#538));
#542=PRODUCT('new_part_id', 'new part name', $, (#220));
#543=PRODUCT_DEFINITION_FORMATION('A', '', #542);
#544=PRODUCT_DEFINITION('sv2', 'design view on sleeve assembly', #538,
#230);
#545=OBJECT_ROLE('output', $);
#546=ROLE_ASSOCIATION(#545, #540);
```

Example 68: exchange file segment for work order

15.2.2 Activity decomposition

The work being directed as part of a work order (the 'work definition') may be further decomposed into sub-activities. These sub-activities are related to the directed_action through an action_relationship (or a chain of such action_relationships) with name 'decomposition'.

When many requests using different solutions are incorporated by a single work order, the directed_action representing the complete work definition should be decomposed in sub-activities for which each appropriate method is indicated.

The items that are affected by a specific sub-activity, e.g. as input or output, may be identified. If the directed_action representing the 'work definition' is broken down in separate sub-activities, it is recommended that this top level work definition be associated through applied_action_assignment to all input and output items that are affected by the sub-activities.

Status information specifying the level of completion of each activity may be specified. Additional administrative information such as persons, organizations, and date and time related information as well as approval information can also be attached to these activities.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements to represent activity decomposition information are illustrated in Diagram 69 below. The corresponding STEP exchange file encoding is given in Example 69.
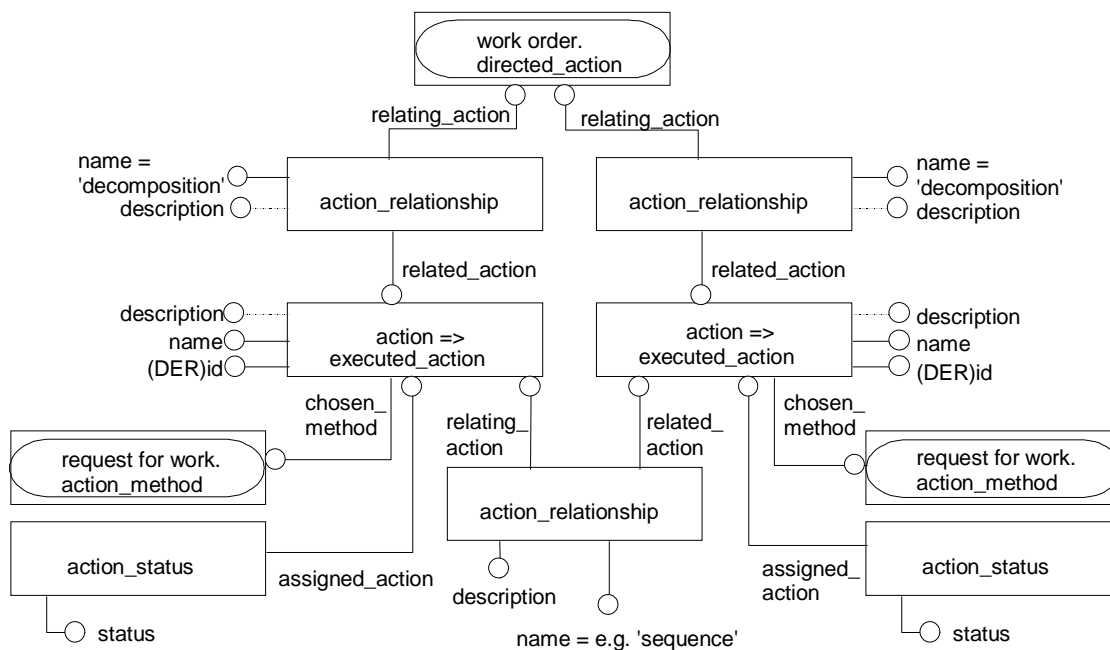

Diagram 69: Activity Decomposition Instance Diagram

15.2.2.1   executed_action

The executed_action entity represents a type of **action** that has been identified as a task that has to be carried out. The activity may be simply identified, partially completed, or completed. An executed_action may have status information associated with it through **action_status** to indicate the level of completeness.

An executed_action refers to an associated action_method that describes the procedure selected to carry out the activity and a description of the results. An executed_action is related to the items that it affects through the applied_action_assignment entity. This is used to identify the inputs and outputs (i.e. results) of the executed_action.

An executed_action should be related directly to a work order as a decomposition of the work definition (directed_action). It is then indirectly related (via the directed_action reference from an action_directive) to the work requests that are addressed by the described activity. By reference to an action_method that is identified as a potential solution to the addressed work_request(s), the executed_action indicates which specific procedure or potential solution for a work_request it has adopted.

Attributes
- The *name* attribute specifies the word or common name used to refer to the action.
- The *description* attribute specifies additional information **about** the action.
- The *chosen_method* attribute specifies the action_method that defines the procedure selected to carry out the **action** and a description of the results of the **action**.
- The *id* attribute is derived, and may store the unique identifier for an executed_action.

ENTITY executed_action	Attribute Population	Remarks
name	type: label = text	
description	type: text = string	OPTIONAL
chosen_method	type: entity = action_method	
id	DERIVEd attribute type: identifier = string	exactly one id may be associated via the entity id_attribute

Preprocessor Recommendations: An executed_action that is not of its subtype directed_action should always be identified as sub-activity in the activity decomposition structure of a directed_action that represents the 'work definition' for a work order. In other words, an executed_action should always be referenced as the related_action in an action_relationship with name 'decomposition' that has a relating_action (potentially through a chain of intermediate action_relationships), which is a directed_action.

It is recommended that an executed_action have an associated status to describe its level of completion, represented by exactly one action_status which references the executed_action as the assigned_action (see 15.2.2.2).

The executed_action entity may have an identifier associated with it through the entity id_attribute and its attribute_value attribute.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An executed_action may be referenced by the following entities:
- action_relationship to associate a directed_action and an executed_action or two executed_actions e.g. to define their decomposition relationship, i.e. one action as part of the other, or to describe their dependencies.
- applied_action_assignment to identify the items that are affected by the executed_action.
- applied_approval_assignment to associate approval information related to an executed_action. See 13.2 for guidance in creating the approval constructs.
- applied_date_assignment (or applied_date_and_time_assignment) to associate date (and time) information to an executed_action. See 13.3.1 for guidance in creating the date and time constructs.
- applied_ organization_assignment (or applied_person_and_organization_assignment) to associate organization (and person) information to an executed_action. See 13.1 for guidance in creating the organization and person constructs. A recommended value for the name attribute of the person_and_organization_role is 'requestor'.
- applied_contract_assignment to identify that an executed_action is subject of the assigned contract.

15.2.2.2   action_status

The action_status entity represents the association of a status with an **executed_action** (or its subtype directed_action) to specify its level of completion.

Attributes
- The *status* attribute specifies the level of completion of the associated executed_action or directed_action.
- The *assigned_action* attribute specifies **the executed_action the status is specified for**.

ENTITY action_status	Attribute Population	Remarks
status	type: label = text	
assigned_action	type: entity = executed_action	

Preprocessor Recommendations: An executed_action shall have at most one associated action_status, which references the executed_action as the assigned_action.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

15.2.2.3   action_relationship

An action_relationship establishes a relationship between two activities represented by directed_action or executed_action instances. The meaning of the action_relationship is given by its name attribute. An action_relationship is used to define an activity as sub-activity, i.e. as part of the decomposition of some higher-level activity. The action_relationship may also be used to express some dependency such as sequential relationship in time, between two activity objects.

In the context of wok management, all defined activities are related either directly or indirectly to a work order. Definition of the complete work to be done, represented by a directed_action, is directly related to a work order. An executed_action defining a sub-activity of the complete work definition is indirectly related to the work order, i.e. related to that directed_action through the action_relationship.

If a directed_action or an executed_action is further decomposed into sub-activities, this shall be a complete decomposition of the higher level activity, i.e. all sub-activities belonging to the same decomposition level together shall completely define the activity on the next higher level.

If the executed_actions defined as sub-activities of a directed_action contain references to items that are either input or output, then these items shall also be explicitly related to the top-level directed_action.

Attributes
- The *name* attribute specifies the meaning of the relationship.
- The *description* attribute specifies additional information text that characterizes the action_relationship.
- The *relating_action* specifies the first action, which is related by the action_relationship. The *relating_action* usually identifies the activity the definition of the related_action is based on, e.g. derived from or dependent on. The semantics of this attribute should be defined by the name attribute.
- The *related_action* specifies the second action related by the action_relationship. The related_action usually identifies the action, which is based on the definition of the relating_action. The semantics of this attribute should be defined by the name attribute.

ENTITY action_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	OPTIONAL
relating_action	type: entity = action	
related_action	type: entity = action	

Preprocessor Recommendations: The name attribute is used to specify the type of the action_relationship. The value 'decomposition' for the name attribute should be used to indicate that the related_action is one of potentially more components into which the relating_action is broken down. Where applicable, the following additional values may be used: 'derivation' to indicate that the related_action is derived from the relating_action; 'precedence' to indicate that the related_action has higher priority than the relating_action; 'sequence' to indicate that the relating_action shall be completed before the related_action starts.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#2=VERSIONED_ACTION_REQUEST('var id2', '0001', 'technical improvement',
'description of var id2, 0001');
#5=ACTION_REQUEST_STATUS('issued', #2);
#9=ACTION_METHOD('technical improvement method 1', $, 'effects of
technical improvement method 1', '$');
#10=ACTION_METHOD('technical improvement method 2', $, 'effects of
technical improvement method 2', '$');
#14=ACTION_REQUEST_SOLUTION(#9, #2);
#15=ACTION_REQUEST_SOLUTION(#10, #2);
#16=APPLICATION_CONTEXT('mechanical design');
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#440));
#220=PRODUCT_CONTEXT('', #16, '');
#230=PRODUCT_DEFINITION_CONTEXT('part definition', #16, 'design');
#340=PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350=PRODUCT_DEFINITION_CONTEXT('assembly definition', #16, '');
#440=PRODUCT('s1', 'sleeve assembly', $, (#220));
#450=PRODUCT_DEFINITION_FORMATION('A', '', #440);
#460=PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly', #450,
#230);
#520=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
#531=APPLIED_ACTION_REQUEST_ASSIGNMENT(#2, (#450));
#534=ACTION_DIRECTIVE('wo id2', 'design change', 'results of analysis
of var id2', 'comment 2', (#2));
#536=DIRECTED_ACTION('design change', $, #10, #534);
#537=ACTION_METHOD('activity method 1', $, 'effects of activity method
1', 'objectives of activity method 1');
#538=PRODUCT_DEFINITION_FORMATION('B', '', #440);
#539=PRODUCT_DEFINITION('sv3', 'analysis view on sleeve assembly',
#538, #230);
#541=APPLIED_ACTION_ASSIGNMENT(#536, (#538, #539, #561));
#548=EXECUTED_ACTION('define requirements', $, #537);
#549=EXECUTED_ACTION('detailed design', $, #551);
#550=EXECUTED_ACTION('analysis and simulation', $, #552);
#551=ACTION_METHOD('activity method 2', $, 'effects of activity method
2', 'objectives of activity method 2');
#552=ACTION_METHOD('activity method 3', $, 'effects of activity method
3', 'objectives of activity method 3');
```

```
#553=ACTION_RELATIONSHIP('decomposition', $, #536, #548);
#554=ACTION_RELATIONSHIP('decomposition', $, #536, #549);
#555=ACTION_RELATIONSHIP('decomposition', $, #536, #550);
#556=ACTION_RELATIONSHIP('sequence', $, #548, #549);
#557=ACTION_RELATIONSHIP('sequence', $, #549, #550);
#558=ACTION_STATUS('completed', #548);
#559=ACTION_STATUS('completed', #549);
#560=ACTION_STATUS('in-work', #536);
#561=PRODUCT_DEFINITION('sv2', 'design view on sleeve assembly', #538,
#230);
#562=APPLIED_ACTION_ASSIGNMENT(#549, (#561));
#563=APPLIED_ACTION_ASSIGNMENT(#550, (#539));
#564=ACTION_STATUS('in-work', #550);
```

Example 69: exchange file segment for activity decomposition

15.3 Project Identification

The project identification area contains the constructs to represent projects, i.e. programs of work for which one or more organizations are responsible, and relationships between projects. Events may be defined to specify when a project is planned to or actually starts and/or ends. Projects may be related to the product concept(s) that are affected by the work carried out within the project

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements to represent project identification information are illustrated in Diagram 70 below. The corresponding STEP exchange file encoding is given in Example 70.
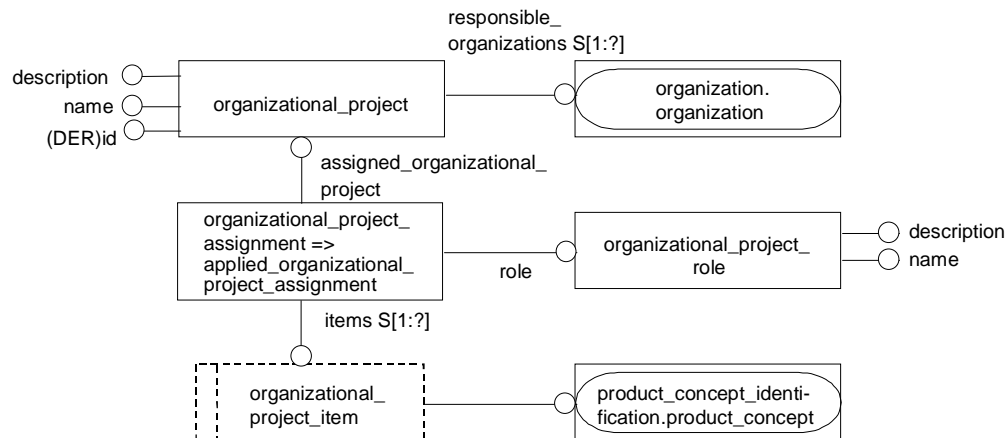
Diagram 70: Project Identification Instance Diagram

15.3.1.1   organizational_project

An organizational_project is an identified program of work for which one or more organizations are responsible.

Attributes
- The *name* attribute specifies the word or common name used to refer to the organizational_project.
- The *description* attribute specifies additional information **about** the organizational_project.
- The *responsible_organization* attribute specifies the organizations that are responsible for the project.

ENTITY organizational_project	Attribute Population	Remarks
name	type: label = text	
description	type: text = string	OPTIONAL
responsible_organization	type: entity = organization	S[1:?]
id	DERIVEd attribute type: identifier = string	exactly one id may be associated via the entity id_attribute

Preprocessor Recommendations: There are no specific preprocessor recommendations.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An organizational_project may be referenced by the following entities:
- organizational_project_relationship to associate two organizational_projects e.g., to define their hierarchical relationships or to describe their dependencies.
- applied_organizational_project_assignment to associate the assigned organizational_project to the product concepts that are affected by the work carried out within the project.
- applied_event_occurrence_assignment to attach event information to organizational_projects. See 13.3.2 for guidance in creating the event_occurrence constructs. Where applicable, the following values for the name attribute of the event_occurrence_role are recommended: 'planned start', 'planned end'.

15.3.1.2   applied_organizational_project_assignment

The applied_organizational_project_assignment entity defines a mechanism to associate an organizational_project to some product data. In the PDM Schema, an organizational_project may be assigned to a product_concept, indicating the product_concept is affected by the work carried out within the assigned organizational_project.

Attributes
- The *assigned_organizational_project* specifies the organizational_project, which is assigned to some product data.
- The *role* identifies the organizational_project_role that specifies the purpose of the association of the **organizational_project_assignment** with product data.
- The *items* attribute specifies the product data the organizational_project is assigned to.

ENTITY applied_ organizational_project_ assignment	Attribute Population	Remarks
assigned_organizational_project	type: entity = organizational_project	
role	type: entity = organizational_project_role	
items	type: entity = organizational_-project_item select	SET[1:?]

Preprocessor Recommendations: The organizational_project may only be assigned to the product_concepts that are affected by the work carried out within the project. The name attribute of the associated organizational_project_role is therefore recommended to be 'affecting project'.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

15.3.1.3   organizational_project_role

An organizational_project_role defines a role for an organizational_project_assignment and a description of that role to characterize the nature of the assignment.

Attributes
- The *name* attribute specifies the word or common name used to refer to the organizational_project_role.
- The *description* attribute specifies **additional information about** the organizational_project_role.

ENTITY organizational_project_role	Attribute Population	Remarks
name	type: label = text	
description	type: text = string	OPTIONAL

Preprocessor Recommendations: An organizational_project may only be assigned to product_concepts that are affected by the work carried out within the project. The name attribute of the associated organizational_project_role is therefore recommended to be 'affecting project'.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An organizational_project_role may be referenced by the following entities:
- applied_organizational_project_assignment to specify the assignment the organizational_project_role defines a role for. The organizational_project_role shall be referenced by at least one applied_organizational_project_assignment.

15.3.1.4   organizational_project_relationship

An organizational_project_relationship is a relationship between two organizational_projects. The meaning of the organizational_project_relationship is given by its name attribute. An organizational_project_-relationship may be used to define an organizational_project as part of some higher-level organizational_project, or to express some dependency such as sequential relationship between two organizational_project objects.

Attributes
- The *name* attribute specifies the meaning of the relationship.
- The *description* attribute specifies additional information text that characterizes the organizational_project_relationship.
- The *relating_organizational_project* specifies the first organizational_project, which is related by the organizational_project_relationship. The relating_organizational_project usually identifies the project the definition of the related_organizational_project is based on, e.g. derived from or dependent on. The semantics of this attribute should be defined by the name attribute.
- The *related_organizational_project* specifies the second organizational_project related by the organizational_project_relationship. The related_organizational_project usually identifies the organizational_project, which is based on the definition of the relating_organizational_project. The semantics of this attribute should be defined by the name attribute.

ENTITY organizational_-project_relationship	Attribute Population	Remarks
name	type: label = string	
description	type: text = string	OPTIONAL
relating_organizational_project	type: entity =	

	organizational_project	
related_organizational_project	type: entity = organizational_project	

Preprocessor Recommendations: The name attribute is used to specify the type of the organizational_project_relationship. Where applicable, the following values should be used:

- 'decomposition' to indicate that the related_organizational_project is a sub-project in a hierarchical breakdown of the relating_organizational_project;
- 'dependency' to indicate that the related_organizational_project is dependent upon the relating_organizational_project; 'sequence' to indicate that the relating_organizational_project must be completed before the related_organizational_project starts;
- 'succession' to indicate that the relating_organizational_project is the successor of the relating_organizational_project.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: There are no specific related entities.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#1=ORGANIZATION('organization_id', 'organization_name', $);
#2=ORGANIZATIONAL_PROJECT('project_name', 'project_description', (#1));
#4=ORGANIZATIONAL_PROJECT_ROLE('affecting project', $);
#5=PRODUCT_CONCEPT('product_concept_id', 'product_concept_name', $,
#7);
#6=APPLIED_ORGANIZATIONAL_PROJECT_ASSIGNMENT(#2, #4, (#5));
#7=PRODUCT_CONCEPT_CONTEXT('pcc_name1', #8, '');
#8=APPLICATION_CONTEXT('');
```

Example 70: exchange file segment for project identification

15.4 Contract Identification

In the STEP PDM Schema, a contract is a binding agreement concerning the design of part versions or the execution of engineering activities. Relationships between contracts may be specified, and the items that are subject of that contract may be associated to it. Additional administrative information such as organization and approval information may also be attached to contracts.

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes essential to support the requirements to represent contract identification information are illustrated in Diagram 71 below. The corresponding STEP exchange file encoding is given in Example 71.
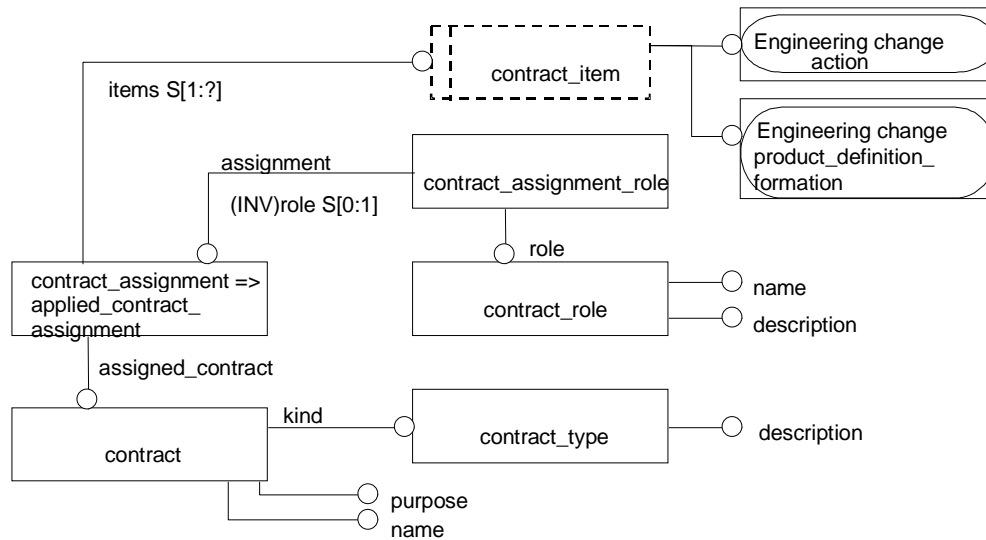
Diagram 71: Contract Identification Instance Diagram

15.4.1.1   contract

The contract entity represents a binding agreement concerning the design of part version, or the execution of engineering activities.

Attributes
- The *name* attribute specifies the word or common name used to identify or refer to the contract.
- The *purpose* attribute is an informal description of the reason for the contract.
- The *kind* attribute specifies the kind of information that a contract conveys.

ENTITY contract	Attribute Population	Remarks
name	type: label = text	
purpose	type: label = text	
kind	type: entity = contract_type	

Preprocessor Recommendations: The mandatory name attribute should contain a string that, in conjunction with the associated contract_type, uniquely identifies the contract.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A contract may be referenced by the following entities:
- applied_contract_assignment to identify the items that are subject of the contract.
- applied_approval_assignment to associate approval information related to a contract. See 13.2 for guidance in creating the approval constructs.
- applied_ organization_assignment to associate organization information to a contract. When assigning organizational info to a contract, the role name of corresponding applied_organization_assignment should be 'organization in contract'. See 13.1 for guidance in creating the organization constructs

15.4.1.2   contract_type

A contract_type is the kind of information that an instance of contract conveys.

Attributes
- The *description* attribute specifies a descriptive label characterizing the type of contract.

ENTITY contract_type	Attribute Population	Remarks
description	type: label = string	

Preprocessor Recommendations: In AP214 the description attribute id used to transfer information on the ordered price of the contracted items.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: A contract_type may be referenced by the following entities:
- contract to identify the contracts that are of that type. A contract_type shall be referenced by at least one contract.

15.4.1.3   applied_contract_assignment

The entity applied_contract_assignment defines a mechanism to associate a contract to the items that are subject of that contract. In the STEP PDM Schema, the binding agreement represented by a contract may be assigned to the design of a part version or to engineering activities that represent the complete work definition of a work order.

The applied_contract_assignment entity may have a role associated with it through the entity object_role. If a role is associated, the action_contract_assignment is referenced as the item_with_role by a role_association where the role is stored in the name attribute of the object_role associated as role.

Attributes
- The *assigned_contract* specifies the contract, which defines a binding agreement with respect to some product data.
- The *items* attribute specifies the items that are subject of the assigned contract.

ENTITY applied_contract_assignment	Attribute Population	Remarks
assigned_contract	type: entity = contract	
role	DERIVEd attribute type: entity = object_role	OPTIONAL
items	type: entity = contract_item select	SET[1:?]

Preprocessor Recommendations: A contract may only be assigned to the part versions (product_definition_formation entities) and engineering activities (directed_action entities) that are subject to the contract. It is recommended to instantiate one object_role associated with the applied_contract_assignment with a name attribute that has the value 'contracted element'.

Postprocessor Recommendations: There are no specific postprocessor recommendations.

Related Entities: An applied_contract_assignment may be referenced by the following entities:
- role_association to specify a object_role defining the purpose of the association of the contract with product data.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
#2=VERSIONED_ACTION_REQUEST('var id2', '0001', 'technical improvement',
'description of var id2, 0001');
#5=ACTION_REQUEST_STATUS('issued', #2);
#9=ACTION_METHOD('technical improvement method 1', $, 'effects of
technical improvement method 1', '$');
```

```
#10=ACTION_METHOD('technical improvement method 2', $, 'effects of
technical improvement method 2', '$');
#14=ACTION_REQUEST_SOLUTION(#9, #2);
#15=ACTION_REQUEST_SOLUTION(#10, #2);
#16=APPLICATION_CONTEXT('mechanical design');
#100=PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#440));
#220=PRODUCT_CONTEXT('', #16, '');
#230=PRODUCT_DEFINITION_CONTEXT('part definition', #16, 'design');
#340=PRODUCT_DEFINITION_CONTEXT_ROLE('part definition type', $);
#350=PRODUCT_DEFINITION_CONTEXT('assembly definition', #16, '');
#440=PRODUCT('s1', 'sleeve assembly', $, (#220));
#450=PRODUCT_DEFINITION_FORMATION('A', '', #440);
#460=PRODUCT_DEFINITION('sv1', 'design view on sleeve assembly', #450,
#230);
#520=PRODUCT_DEFINITION_CONTEXT_ASSOCIATION(#460, #350, #340);
#531=APPLIED_ACTION_REQUEST_ASSIGNMENT(#2, (#450));
#534=ACTION_DIRECTIVE('wo id2', 'design change', 'results of analysis
of var id2', 'comment 2', (#2));
#536=DIRECTED_ACTION('design change', $, #10, #534);
#538=PRODUCT_DEFINITION_FORMATION('B', '', #440);
#541=APPLIED_ACTION_ASSIGNMENT(#536, (#538));
#544=CONTRACT('contract_name', 'purpose of contract', #545);
#545=CONTRACT_TYPE('description of contract type');
#546=APPLIED_CONTRACT_ASSIGNMENT(#544, (#536, #538));
#547=OBJECT_ROLE('contracted element', $);
#548=ROLE_ASSOCIATION(#547, #546);
```

Example 71: exchange file segment for contract identification

15.5 Known issues

15.5.1 Person & Organizations signing for a contract

The PDM Schema currently does not support attaching information indicating which the person and organizations signed a contract. This capability requires an extension of the select type organization_item by applied_organization_assignment.

15.5.2 Associating activities to projects

The PDM Schema currently does not support associating actions to an organizational_project to identify which activities are considered part of a given project. This capability requires the extension of the select type organizational_project_item to include the type action.

16 Measure and units

A measure with unit is the specification of a physical quantity as defined in ISO31 (clause 2). The PDM Schema allows specifying different measure with units used for, e. g., a property definition or quantity of an assembly relationship. A measure with unit is characterized by two components, ; a value specifying the quantity and a unit in which the value is expressed.

The PDM Schema supports the application of:
*   simple units including predefined SI units according to ISO1000 (clause 2);
*   converted units which are derived from another unit by a conversion factor;
*   derived units which are defined by an expression of units;
*   user defined units which are dependent on a specific context and which are not related to the system of units within the PDM Schema.

16.1 Measure with unit specification

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of measure with unit are shown in Diagram 72.
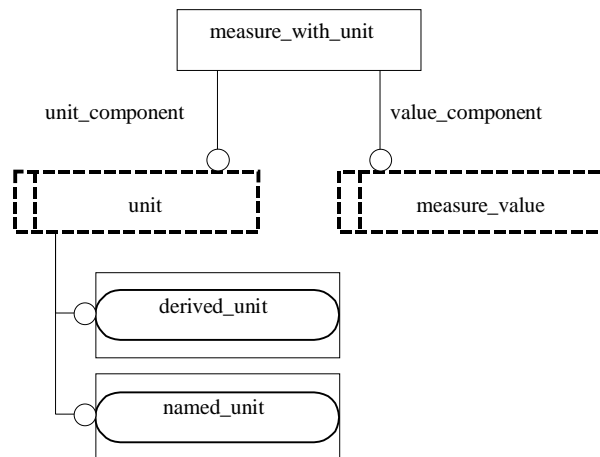
Diagram 72 : Measure with Unit Entity and Attributes

16.1.1.1   measure_with_unit

A measure_with_unit is the specification of a physical quantity as defined in ISO31 (clause 2). It may be of subtype measure_representation_item if it is used in the context of properties associated with product data, e. g., to represent the value and unit of a propertyor it may be of subtype uncertainty_measure_with_unit .

A measure_with_unit may be typed by one of the following predefined subtypes depending on the used unit definition (16.2), if it does not represent a derived unit (16.2.3):

SUPERTYPE OF (ONE OF (length_measure_with_unit, mass_measure_with_unit,
time_measure_with_unit, electric_current_measure_with_unit,
thermodynamic_temperature_measure_with_unit, amount_of_substance_measure_with_unit,

luminous_intensity_measure_with_unit, plane_angle_measure_with_unit, solid_angle_measure_with_unit, area_measure_with_unit, volume_measure_with_unit, ratio_measure_with_unit));

Attributes
- The ***unit_component*** attribute provides the unit in which the physical quantity is expressed.
- The ***value_component*** attribute provides the value of the physical quantity if expressed with respect to the unit_component.

ENTITY measure with unit	Attribute Population	Remarks
unit_component	type: unit = named_unit, derived_unit	references associated unit_component
value_component	type: measure_value = length_measure, mass_measure, time_measure, electric_current_measure, thermodynamic_temperature_measure, amount_of_substance_measure, luminous_intensity_measure, plane_angle_measure, solid_angle_measure, area_measure, volume_measure, ratio_measure, parameter_value, numeric_measure, context_dependent_measure, descriptive_measure, positive_length_measure, positive_plane_angle_measure, positive_ratio_measure, count_measure	references associated value_component; the type of measure_value is constrained by the selected subtype of measure_with_unit

Preprocessor Recommendations:
It is recommended to use one of the predefined subtypes to precise measure_with_unit if it does not represent a derived_unit and if it is applicable. The type of measure_value is constrained by the respective subtype of measure_with_unit, e. g. mass_measure is required if mass_measure_with_unit is applied.

Postprocessor Recommendations: None specific

Related Entities: A measure_with_unit may be referenced by the following entities:
- conversion_based_unit - as the *conversion factor* that specifies the physical quantity from which the conversion_based_unit is derived
- lot_effectivity - for the specification of the *effectivity lot size*, i.e. the size of the batch of items
- make_from_usage_option - for the specification of the amount of the relating_product_definition that can be made from the related_product_definition
- quantified_assembly_component_usage - for the specification of a measure that defines how many or how much of the constituent is used in the assembly

16.2 Unit definition

16.2.1 Simple and predefined units

A simple unit is a named unit where the type of unit is defined only implicitly by the selected type of measure_value for value_component (table in 16.1.1.1) and not explicitly by using any predefined subtype of named_unit. Examples are count_measure like "each". .

A predefined unit is a named unit where the unit type is defined by using a predefined subtype of named_unit. Examples are length_unit or mass_unit. A special type of a predefined unit is an SI unit based on the specifications of ISO1000 (clause 2).

The Instance Model: EXPRESS entities and attributes

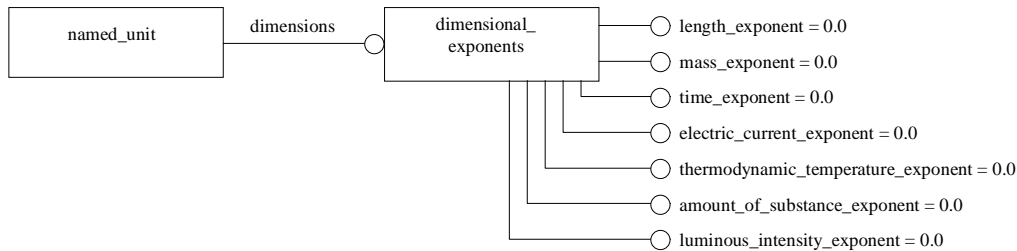The EXPRESS entities and attributes used to support the requirements of named unit are shown in Diagram 73.

Diagram 73: Named Unit Instance Diagram

An instantiation example is shown in Diagram 74 representing a predefined length unit "millimeter".
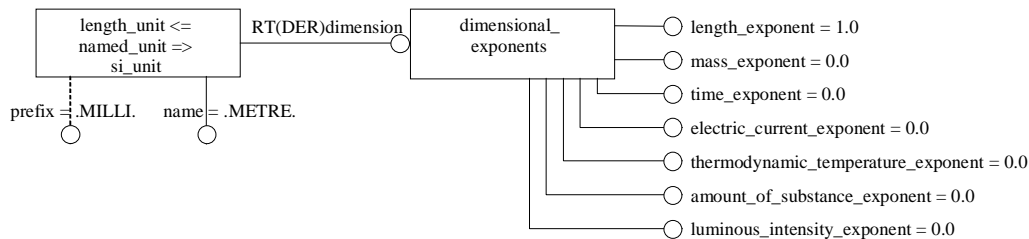
Diagram 74: SI Unit Instance Diagram

16.2.1.1   named_unit

A named_unit is a unit quantity. A named_unit may be of subtype si_unit (16.2.1.3), conversion_based_unit (16.2.2.1) or context_dependent_unit (16.2.4.1).

A named_unit may be typed (also) by one of the following predefined subtypes:

SUPERTYPE OF (ONEOF(si_unit, conversion_based_unit, context_dependent_unit) ANDOR ONEOF(length_unit, mass_unit, time_unit, electric_current_unit, thermodynamic_temperature_unit, amount_of_substance_unit, luminous_intensity_unit, plane_angle_unit, solid_angle_unit, area_unit,volume_unit, ratio_unit))

The attribute dimensions specifies the powers of the dimensions of seven base quantities via dimensional_exponents (16.2.1.2). If named_unit is of subtype si_unit, there is a rule specifying how to instantiate the dimensional_exponents attributes referenced via the derived dimensions attribute of si_unit.

Attributes
- The *dimensions* attribute provides the dimensional_exponents of the base properties by which the named_unit is defined.

ENTITY named_unit	Attribute Population	Remarks
dimensions	type: entity = dimensional_exponents	references associated dimensional_exponents

Preprocessor Recommendations: In the case of a simple unit the predefined subtypes (length_measure, mass_measure etc.) are not applicable. Therefore named_unit is instantiated by its own.

In all other cases (wherever applicable) it is recommended to use one of the predefined subtypes to precise a named_unit. It should be done via a complex instance with one of the other subtypes of named_unit (context_dependent_unit, conversion_based_unit or si_unit). It is recommended to not use a predefined subtype by its own without combining it with context_dependent_unit, conversion_based_unit or si_unit.

The complex instance is also recommended for si_unit (even though a predefined si_unit should be of common understanding world wide) to be compatible as it has always been implemented, but with the exceptions of area_unit and volume_unit (see 16.2.1.3).

The predefined subtypes of named_unit are constraint by respective subtypes of measure_with_unit, e. g. mass_unit is required, if mass_measure_with_unit is applied.

Postprocessor Recommendations: None specific

Related Entities: A named_unit may be referenced by the following entities:
- measure_with _unit - as the *unit component* that provides the unit in which the physical quantity is expressed
- derived_unit_element - as the *unit* specifying the mathematical factor of the element

16.2.1.2   dimensional_exponents

The dimensional_exponents entity defines the powers of the dimensions of the seven base quantities.

Attributes
- The *length_exponent* attribute provides the power of the length base quantity
- The *mass_exponent* attribute provides the power of the mass base quantity
- The *time_exponent* attribute provides the power of the time base quantity
- The *electric_current_exponent* attribute provides the power of the electric current base quantity
- The *amount_of_substance_exponent* attribute provides the power of theamount of substance base quantity
- The *luminous_intensity_exponent* attribute provides the power of luminous intensity base quantity
- The *thermodynamic_temperature_exponent* attribute provides the power of the thermodynamic temperature base quantity

ENTITY dimensional_exponents	Attribute Population	Remarks
length_exponent	type: REAL	
mass_exponent	type: REAL	
time_exponent	type: REAL	
electric_current_exponent	type: REAL	
amount_of_substance_exponent	type: REAL	
luminous_intensity_exponent	type: REAL	
thermodynamic_temperature_exp onent	type: REAL	

Preprocessor Recommendations:
The values of the single attributes of dimension are constraint by where clauses and function in PDM Schema dependent on the selected subtype of named_unit.

Postprocessor Recommendations: None specific

Related Entities: None specific.

16.2.1.3 si_unit

An si_unit is a subtype of named_unit that defines a unit with respect to the system of units defined in this schema. The system of units is based on the specifications of ISO1000 (clause 2) but differs from them for the unit of mass.

The Instance Model: EXPRESS entities and attributes

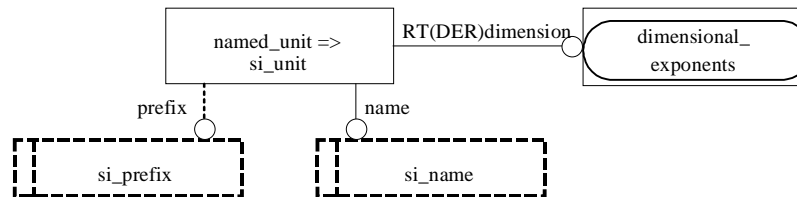The EXPRESS entities and attributes used to support the requirements of SI unit are shown in Diagram 75.

Diagram 75: SI Unit Entities and Attributes

Attributes
- The *dimensions* attribute provides the dimensional_exponents for the si_unit (see 16.2.1.2)
- The *name* attribute provides the name of an SI unit. The definitions of names of SI units are specified in ISO1000 (clause 2)
- The *prefix* attribute provides the name of a prefix that may be associated with an si_unit. The definitions of SI prefixes are specified in ISO1000 (clause 3).

ENTITY si_unit	Attribute Population	Remarks
dimensions	type: entity = dimensional_exponents	retransmitted, DERIVED reference to the associated dimensional_exponents
name	type: si_unit_name = (metre, gram, second, ampere, kelvin, mole, candela, radian, steradian, hertz, newton, pascal, joule, watt, coulomb, volt, farad, ohm, siemens, weber, tesla, henry, degree_celsius, lumen, lux, becquerel, gray, sievert);	the name attribute indicates the name of the represented si_unit
prefix	type: si_prefix = (exa, peta, tera, giga, mega, kilo, hecto, deca, deci, centi, milli, micro, nano, pico, femto, atto)	OPTIONAL

Preprocessor Recommendations: A complex instance with one of the predefined subtypes of named_unit is recommended for si_unit (even though si_unit should be of common understanding world wide) to be compatible as it has always been implemented (see examples in Example 71). However, area_unit and volume_unit are not allowed in combination with si_unit.

It is recommended to not use any prefix if si_unit is referenced by conversion_based_unit.conversion_factor (via measure_with_unit.unit_component) to simplify the calculation of the conversion by using only a basic SI unit.

Postprocessor Recommendations: None specific

Related Entities: None specific.

16.2.2 Converted units

16.2.2.1   conversion_based_unit

A conversion_based_unit is a type of named_unit that defines a unit on the basis of a measure_with_unit. The value_component of the measure_with_unit defines the conversion factor. An inch is an example for a conversion based unit .

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of conversion based unit are shown in Diagram 76.
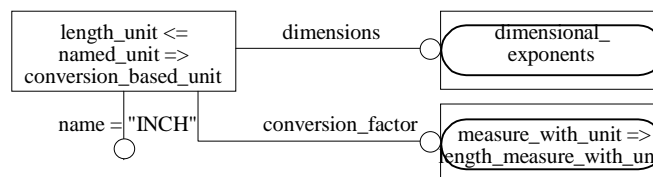

Diagram 76: Conversion Based Unit Instance Diagram

Attributes
- The *dimensions* attribute provides the dimensional_exponents for the conversion_based_unit (see 16.2.1.2)
- The *name* attribute provides the label by which the conversion_based_unit is known.
- The *conversion_factor* attribute provides the measure with unit that specifies the physical quantity from which the conversion_based_unit is derived.

ENTITY measure with unit	Attribute Population	Remarks
dimensions	type: entity = dimensional_exponents	references to the associated dimensional_exponents
name	type: label = STRING	the name attribute indicates the name of the represented conversion_based_unit
conversion_factor	type: entity_measure_with_unit	references the associated conversion factor

Preprocessor Recommendations: If applicable a complex instance with one of the predefined subtypes of named_unit is recommended for conversion_based_unit (see examples in Example 71).

If applicable only basic measure with units (e. g. SI units without prefix) should be used for conversion_factor (via measure_with_unit.unit_component). It is recommended to simplify the calculation of the conversion by avoiding chains of conversions.

Postprocessor Recommendations: None specific

16.2.3 Derived units

16.2.3.1   derived_unit

A derived_unit is an expression of units. For instance "Newton per square millimeters" is a derived_unit. A derived_unit is also typically used to specify a surface or volume unit as derivation from a length unit like "square meter" derived from "meter".

The predefined subtypes of measure_with_unit are not applicable for a derived_unit, because a derived_unit may be expressed by measure with units of different (sub)types like "Newton per square millimeters".

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of derived unit are shown in Diagram 77.
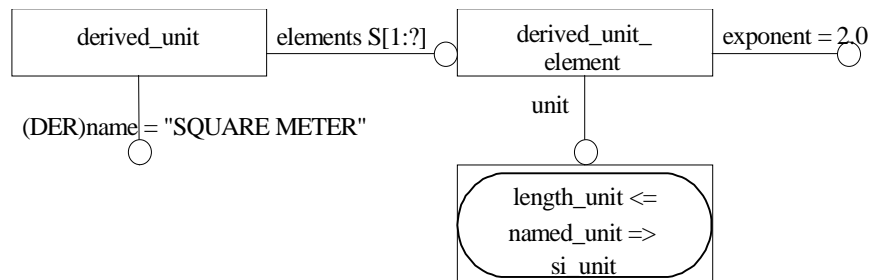
Diagram 77: Derived Unit Instance Diagram

Attributes
- The *name* attribute provides the label by which the derived_unit is known.
- The *elements* attribute provides the derived_unit_element objects and their exponents that define the derived_unit.

ENTITY derived_unit	Attribute Population	Remarks
Name	type: label = STRING	DERIVED attribute which indicates the name of the represented derived_unit
Elements	type: entity = S[1:?] of derived_unit_element	references the associated elements of the unit expression

Preprocessor Recommendations: None specific

Postprocessor Recommendations: None specific

Related Entities: None specific.

16.2.3.2   derived_unit_element

A derived_unit_element is the association of a named_unit with an exponent. This entity is used to represent an element of the dimensional expression of a derived_unit. For instance "Newton per square millimeter" is a derived_unit. It has two elements, "Newton" whose exponent has a value of 1.0 and "millimeter" whose exponent is -2.0.

Attributes
- The *unit* attribute provides the named_unit that specifies the mathematical factor of the element.
- The *exponent* attribute provides the power that is applied to the unit attribute.

ENTITY derived_unit_element	Attribute Population	Remarks
Unit	type: entity = named_unit	references the associated named_unit
Exponent	type: REAL	identifies the exponent of the referenced named_unit

Preprocessor Recommendations: None specific

Postprocessor Recommendations: None specific

Related Entities: None specific.

16.2.4 User defined units

A user defined unit is a named unit which can neither be described as a simple unit, SI unit, a converted unit nor a derived unit. A user defined unit usually depends on a specific context the unit is used in, e. g., if a physical quantity depends on a global unit specified for the referenced object. The entity context_dependent_unit supports the representation of a user defined unit.

16.2.4.1   context_dependent_unit

A context_dependent_unit is a type of named_unit which is not related to the system of units defined in this standard. It should only be used if neither si_unit, conversion_based_unit, derived_unit nor the predefined subtypes of named_unit are applicable.

A context_dependent_unit should also be used, if the physical quantity depends on a global unit specified for the referenced object or if the physical quantity is not described explicitly, but depends on a specific context. For example the unit may be "one milliliter" or "one gram" of lubricant depending on the global unit specified for the lubricant being "milliliter" or "gram" in the case of a quantified assembly relationship with quantity "one each".

The Instance Model: EXPRESS entities and attributes

The EXPRESS entities and attributes used to support the requirements of context dependent unit are shown in Diagram 78.
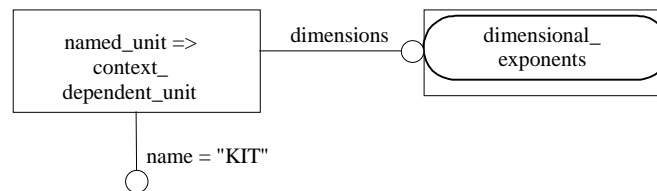
Diagram 78: Context Dependent Unit Instance Diagram

Attributes
- The *dimensions* attribute provides the dimensional_exponents for the context_dependent_unit (see 16.2.1.2).
- The *name* attribute provides the label by which the context dependent unit is known.

ENTITY context_dependent_unit	Attribute Population	Remarks
Dimensions	type: entity = dimensional_exponents	references associated dimensional_exponents
Name	type: label = STRING	the name attribute indicates the name of the represented context_dependent_unit

Preprocessor Recommendations: If applicable a complex instance with one of the predefined subtypes of named_unit is recommended for context_dependent_unit.

Postprocessor Recommendations: None specific

Related Entities: None specific.

The Instance Model: STEP exchange file format (ISO10303 Part 21 syntax)

```
/* Millimeter [mm]: */
#1=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(1.0),#2);
#2=(LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.MILLI.,.METRE.));

/* Gram [g]: */
#5=MASS_MEASURE_WITH_UNIT(MASS_MEASURE(150.5),#6);
#6=(MASS_UNIT() NAMED_UNIT(*) SI_UNIT($,.GRAM.));

/* Square meter [m2]: */
#10=AREA_MEASURE_WITH_UNIT(AREA_MEASURE(2.5E+001),#20);
#20=DERIVED_UNIT((#30));
#30=DERIVED_UNIT_ELEMENT(#50,2.0);
#40=NAME_ATTRIBUTE('SQUARE METER',#20);
#50=(LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));

/* Inch [in]: */
#110=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.54E-002),#120);
#120=(CONVERSION_BASED_UNIT('INCH',#140) LENGTH_UNIT()
NAMED_UNIT(#130));
#130=DIMENSIONAL_EXPONENTS(1.0,0.0,0.0,0.0,0.0,0.0,0.0);
#140=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(5.0),#150);
#150=(LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));

/* Newton per square millimeter [N/mm2]: */
#201=MEASURE_WITH_UNIT(NUMERIC_MEASURE(500.),#202);
#202=DERIVED_UNIT((#203,#204));
#203=DERIVED_UNIT_ELEMENT(#205,1.);
#204=DERIVED_UNIT_ELEMENT(#206,-2.);
#205=SI_UNIT(*,$,.NEWTON.);
#206=(LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.MILLI.,.METRE.));

/* Liter [l]: */
#300=VOLUME_MEASURE_WITH_UNIT(VOLUME_MEASURE(1.5),#310);
#310=(CONVERSION_BASED_UNIT('LITER',#330)NAMED_UNIT(#320)
VOLUME_UNIT());
#320=DIMENSIONAL_EXPONENTS(3.0,0.0,0.0,0.0,0.0,0.0,0.0);
#330=LENGTH_MEASURE_WITH_UNIT(VOLUME_MEASURE(1.0E-003),#340);
#340=DERIVED_UNIT((#350));
#350=DERIVED_UNIT_ELEMENT(#360,3.0);
```

```
#360=(LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));

/* Each []: */
#400=MEASURE_WITH_UNIT(COUNT_MEASURE(2),#401);
#401=NAMED_UNIT(#402);
#402=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);

/* Unit of prefabricated []: */
#500=MEASURE_WITH_UNIT(CONTEXT_DEPENDENT_MEASURE(3.0),#501);
#501=CONTEXT_DEPENDENT_UNIT(#502,'UNIT OF  PREFABRICATED');
#502=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);

/* As required []: */
#600=MEASURE_WITH_UNIT(CONTEXT_DEPENDENT_MEASURE(1.0),#601);
#601=CONTEXT_DEPENDENT_UNIT(#602,'AS REQUIRED');
#602=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);
```

Example 72: Examples of mapping different units

Annex A   PDM Schema EXPRESS listing

A.1 PDM Schema short form

http://www.pdm-if.org/pdm_schema/pdm11_sf.exp

A.2 PDM Schema long form

http://www.pdm-if.org/pdm_schema/pdm11lf.exp

Annex B   PDM Schema EXPRESS-G Diagrams

http://www.pdm-if.org/pdm_schema/pdm11-exg.pdf

Annex C   PDM Schema Issues Log

http://www.pdm-if.org/pdm_schema/pdm11-issues_log.pdf

Annex D   Document Change Log

Changes to Release 1.0
- Removal of creation information recommendation in section 1.1.1,
- Removal of creation information recommendation in 1.1.2,
- Addition of Document Identification (section 5),
- Addition of File Identification (section 6),
- Addition of Authorization - Person and Organization (section 13.1),
- Addition of Authorization - Approval (section 13.2).

Changes to Release 2.0
- See Closed Issues in the associated PDM Schema Usage Guide Issue Table

Changes to Release 3.0
- Addition of approval cycle descriptions and approval_relationship in section 13.2.2,
- Moved sections 2.4 and 2.5 (relating shape to product structure) to sections 4.4 and 4.5,
- Added section 2 (specific part type classification),
- Added section 6 (specific document type classification),
- Added section 12 (alias identification),
- Added section 13.3 (date, time, and event reference),
- Added section 14 (configuration and effectivity information),
- Added section 15 (engineering change and work management).

Changes to Release 4.0
- Added section 16 (measure and units),
- Additions to section 4 (drawings, parts lists, and item find number),
- Modifications to section 9 (document properties),
- Numerous updates and fixes (refer to issue log).

Index

PDM SCHEMA USAGE GUIDE – ISSUES LOG

issue number	author	classification	status	affected release	abstract	description	proposed solution
71	AP203	technical	closed	2.0	assignment of 'creator' to product entity	AP203 and related implementation pilots have no requirement for identification of the 'creator' for the product entity (base of the part master) - requirement for creator of part version (product_definition_formation).	remove recommendation for 'creator' assigned to product entity - [edited table in section 13.1 - table of assignment recommendations]
33	AP214	major tech	closed	2.0	ddid.relevant_for	the approach linking product_definitions to application_contexts in the UG does not conform with the 214 mapping that requires a product_definition_context_association. the product_definition_context referenced by frame_of_reference in 214 is not a life cycle	the approach linking product_definitions to application_contexts in the UG does not conform with the AP214 mapping that requires a product_definition_context_association. This is listed as a known issue in the usage guide, and is an issue against AP214 DIS. [added known issues text in usage guide, generated AP214 DIS ballot comment]
51	BaE	technical	closed	2.0	Use of document_file.id, document_file.name, document.id	On document_file. Should document.name be the file name? E.g. the word document? The UG is not clear on this. Also, what is document.id used for?	For external files, document_file.name contains the nomenclature for the file being referenced. For 'document as product', product.name should contain the document name. [clarified in section 9.7]
56	BaE	technical	closed	2.0	Confusing description of document and file association to product data	should NOT use product_definition_with_associated_documents. Therefore we should just use applied_document_reference. We don't need to use document_product_equivalence???	clarify description of document/file association to product data - clarify section 10 [added text to clarify]
59	BaE	technical	closed	2.0	Distinction between a file and a document in a PDM system (see Issue #24)	What is the difference between a document and a document_file?	[Clarified in sections 7 and 10]
63	BaE	technical	closed	2.0	Assignment of person/organization to part/document master	just use applied_person_and_organization_assignment with product? If we use applied_person_and_organization_assignment with product_definition_formation, What roles should we use? Are any of the roles mandatory (the use of 'id owner' is only recommended in section 1.1.1.1)?	in general, the pdm schema has very few EXPRESS mandatory contraints - the usage guide attempts to provide recommendations to guide consistent use and implementation. Any explicitly recommended string should be used in order to maximize the chances for interoperability. Other values that are defined but not explicitly recommended should be used as defined when appropriate to help facilitate potential interoperability. See table 3 in section 13.1 [added text to clarify - explain additional role strings that may be applicable]
36	BaE	technical	closed	2.0	Product_definition mandatory for each part? (see Issue #2)	Can a product just have a version and no view i.e., just product with associated product_definition_formation?	yes, it is possible in restricted use. in general, each product_definition should have an associated product_definition representing the view definition for the part version. In two specific cases a part version may be exchanged without an associated product_definition_formation should have an associated product_definition. 1 - where version history is represented - only the most recent version is assigned a product_definition. 2 - where supplied item is identified using the alias relationship - the supplier part version may not have an associated product_definition. Clarify section 1.1.1.2 [added test to describe the two specifc cases, explained that versions without definitions are not to describe those versions, but to enhance information about related defined versions]
38	BaE	technical	closed	2.0	Use of product_definition.id? (see Issue #37)	What are allowable views in product_definition.id? (see Issue #37)	(see solution Issue #37)
39	BaE	technical	closed	2.0	Product_definition_formation_with_specified_source	Why is product_definition_formation_with_specified_source not recommended - where do we put the 'make' or 'buy' information?	the make vs. buy distinction can be ambiguous in the context of exchange - going down a supply chain, the sender is typically 'buy' while the receiver is 'make'. In the other direction, the sender is often 'make' and the receiver 'buy' - because of this ambiguity recommendations for exchange are generally to not use this subtype. Clarify section 1.1.1.3 [added text to clarify]
41	BaE	technical	closed	2.0	Product_definition_context_association for documents	We are assuming that product_definition_context_association is only required for parts and not for documents. Is this assumption correct?	yes - Clarify section 1.1.2.5 [added text to clarify]

issue number	author	classification	status	affected release	abstract	description	proposed solution
42	BaE	technical	closed	2.0	Definition of 'detail' for part type	Regarding "Type Classification", what is a detail and how is it different to a part?	a 'detail' is a 'part' - it represents a part that is not broken down into other identified, managed component parts in a product structure. 'detail' is used to distinguish from 'assembly' if this is required when exchanging product structure. Clarify section 1.1.3. [added text to clarify]
46	BaE	technical	closed	2.0	Use of product_definition.id for document as product (see Issues #4, #5)	What value should be given for product_definition.id when using document as product?	unique identification of the definition of the document view/representation - clarify section 5.1.2.1 [added text to clarify].
57	BaE	technical	closed	2.0	Why applied_document_reference to product_definition_formation_relationship?	Why would you want to relate a document to a relationship?	the requirement to assign a document (or file) to an individually identified occurrence of a 'definitional template' for a part - reference section on part structure - clarify section 10.1.1.4 [added text to clarify]
58	BaE	technical	closed	2.0	Semantics of object role with applied_document_reference (see Issue #23)	necessary to associate a role (object_role) with the applied_document_reference. The two permitted values for object_role.name are 'mandatory' and 'informative'. What do these mean?	'mandatory' means the assignment of the document to the product data is in a mandatory relationship - the document must be taken into account to understand the complete product information. 'informative' means the assignment of the document to the product data is an optional relationship - the document may be considered for additional information but is not required or enforced. clarify section 10.1.1.6 (see issue #23) [added text to clarify].
32	DASA (EF)	technical	closed	2.0	issue against product_requires_version	application projects want to exchange part definitions (products) without versions	this basic identification structure has been a central concept for version control and configuration management in AP203 - also there is an EXPRESS rule enforcing the requirement to have product_definition_formation required for product. other projects that have not seen version identification as an exchange requirement have used various strings to indicate a "dummy" version : 'ANY' - in the context of a product structure where the implied semantic was 'current/latest' version; 'NULL' - with no version semantic implied/supplied. as an aside, AIAG studies have shown that a major cost factor in automotive supply chains results from rework due to incorrect version information. Clarify section 1.1.1.2 [added text clarifying the requirement]
34	DASA (EF)	technical	closed	2.0	document_file.description valid for user data? (see Issue #1)	clarify if document_file.description is valid for user data.	Document_file.description is an example of an optional description attribute, available on many entities. In general, use of this type of attribute is given the following recommendation: preprocessor - first, follow the usage guide as much as is possible - if some specific common harmonized user requirement has been documented in the usage guide for the attribute, try to adapt this requirement to those you have identified. If no specific common harmonized user requirement has been documented in the usage guide, a general characterization of the attribute is given in the usage guide description of the entity. In this case, the optional description attribute should not be instantiated. However, this type of attribute may be used in some bilateral agreements between exchange partners. When this is done, the use of the attribute should be in line with the general characterization of the attribute given in the usage guide. These agreements, though, will not generally be interoperable solutions. [added text to general information section]
2	PDM Editor	editorial	closed	2.0	product_definition mandatory for each part??	Currently the Ug may lead to the interpretation that each part requires a product_definition	clarify that this is not the case (see solution Issue #36)
3	PDM Editor	editorial	closed	2.0	values for product_related_product_category.name (see Issue #35)	The usage guide is not clear whether e.g. 'tool' or 'raw material' can be used here	clarify, product_related_product_category is revisited for materials clarify that the section refers to item classification of parts (see solution Issue #35)
4	PDM Editor	technical	closed	2.0	Semantics of product_definition.id for document representations	conflict between 214 and UG: 'identification' vs 'type of the document …'	seems like a typo and should be corrected as 214 states (see solution Issue #46)
5	PDM Editor	technical	closed	2.0	correspondance of document_representation.id to product_definition.id not stated	product_definition.id is said to have no standard mapping. AP214 in fact maps document_representation.id here	rephrase to state correspondance to id of document representation (see solution Issue #46)
7	PDM Editor	technical	closed	2.0	uniqueness of document_file.id	the ug does not state that document_file.id shall be unique. but for file identification this is required	state uniqueness - clarify section 7.1.1.1 [added text to section 7.1.1.1]
8	PDM Editor	editorial	closed	2.0	document_file.id	not document_file.id but document_file.description is double inherited	correct [see solution Issue #53]

issue number	author	classification	status	affected release	abstract	description	proposed solution
9	PDM Editor	major tech	closed	2.0	organization.type	organization.type is mandatory in 214 ARM. The mapping is to organization.description. Description is optional in the UG	issue against AP214 DIS to make organization.type optional [no change to usage guide]
11	PDM editor	technical	closed	2.0	date_and_time	recommendations for standardised date format should be given. (year 4 digits)	add recommendations section 13.2 [text added for four digits]
13	PDM Editor	editorial	closed	2.0	multiple organisation assignment to one approval (see Issue #28)	it is not made clear that multiple org can be assigned to one approval	clarify section 13.2 [added text to hopefully clarify]
14	PDM Editor	technical	closed	2.0	approval_role.description	the ug requires approval_role.description to be 'approver'. This conflicts with 214 Rec Prac that asks for an empty string	the usage guide actually recommends 'approver' for the approval_role attribute, not approval_role.description.
15	PDM Editor	technical	closed	2.0	approval_date_time.role	The Ug does not give hints on the instantiation of this attribute. Should corrsp. to 214 be 'planned' or 'actual'. Are both - planned_date and actual_date - in scope? Or is it just the date of the sign-off?	for any given approval, the approval_date_time.role is intended as 'actual' - the date the current status value was actually assigned - clarify section 13.2 [added text to clarify]
1	PDM Editor	major tech	closed	2.0	guidance on the usage of AIM attributes for which no standard mappings exists	The usage guide describes sematics for a number of Aim attributes for which no standard mapping exists (in the interesection of the APs). E.g. person,prefix_titels, person_and_organization.description	Add a general section in the usage guide stating that optional attributes shall in general not be instaniated unless otherwise indicated that they are valid for user data. Add health warning. Mark manadatory attributes that shall not be instaniated explic [see solution Issue #34]
29	PDMI2	technical	closed	2.0	URL notation for a unique organization id	an approach to a unique identification of organisation is needed and shall be recommended	suggest that the use of a URL-like notation such as prostep.de may be useful if appropriate - clarify section 13.1 [added text to clarify]
35	STAMP	technical	closed	2.0	Use of 'tool' and 'raw material' (see Issue #17)	The latest RPG for PDM schema (May 27) is a little vague on the product_related_product_category's for "raw material" and "tool".	A primary requirement for product_related_product_category is to distinguish 'part' from 'document'. For upward compatibility, 'detail', 'assembly', and 'standard' are also values recognized for product_related_product_category (in addition and related to 'part'). These may be important basic type requirements in the context of a part assembly structure. The value 'tool', as opposed to 'part' or 'raw material', identifies a product that is ultimately employed in the role of manufacturing tooling for use in the production of an actual part. However, this value may be subject to the same type of ambiguity in exchange as 'make' vs. 'buy' information - reference Issue #39. It depends upon which end of the exchange you are on e.g. the supplier's part may be the buyer's tool. However, consistent use of the above definition may help resolve the ambiguity. [deleted text referring to 'raw material' and 'tool'].
43	STAMP	technical	closed	2.0	Drawings - how are drawings represented in PDM schema?	will a drawing be instantiated as a PRODUCT ?	yes - a configuration controlled managed document like a drawing would generally map to 'document as product' (with a physical representation/view definition). Clarify section 5.0 [added text in example]
45	STAMP	technical	closed	2.0	Can we assume that DOCUMENT.id will equal the document PRODUCT.id ?	Use Product.id, not Document.id to identify document? Should be same? Document does not always need to be instantiated with the DOCUMENT entity?	Use Product.id, not Document.id to identify document. Document.id should not be used as valid user data. The representation of a document as product does not always need to be instantiated with the document entity - this is done only to assign the document to other product data via applied_document_reference. Clarify section 5.1.1, 10.1.1.2. [added text to 5.0 and 10.1.1.2 pre- and post-processor recommendations]
48	STAMP	technical	closed	2.0	General issue of what is controlling te configuration of the file (see Issue #20)	The part the file is pointing to? - or is it independently controlled (which means it is a document itself, with its own meta data in the PDM system)?	agree - if a file is under configuration control, it is related as a constituent of a document view/representation definition and it is actually represented by a change to the managed document that is under direct configuration control. In this case, a change to a file is actually represented by a change to the managed document (i.e. a new version) - the changed file would be mapped as a constituent of a view/representation definition of the new document version. A simple external reference (a non managed file not under configuration control) may point to product data, but i don't think the configuration control of the referenced product data necessarily applies to the file. This file is not configuration controlled, it is simply an externally reference to product data. Clarify section 5, 7, 8 [added paragraph in section 5, 7, and 8].
60	STAMP	technical	closed	2.0	Distinction between a file and a document in a PDM system (see Issue #59)	The underlining issues is what is the distinction between a file and a document in a PDM system. Hopefully the distinction is the same in all commercial PDM systems. In MetaPhase ...	That is a Metaphase issue. The ability to handle non-managed files is required by users and supported by other PDM systems

issue number	author	classification	status	affected release	abstract	description	proposed solution
61	STAMP	technical	closed	2.0	Relating files to parts directly (see Issue #24)	Could we use APPLIED_DOCUMENT_REFERENCE to relate a file to a part or document as opposed to going the route of PROPERTY_DEFINITION_REPRESENTATION.etc?	When relating a CAD file to a part without stating location information, use applied_document_reference. Always use applied_document_reference when relating a file to product data. If the file contains the definition of the part, use property_def_rep and shape_rep in addition to applied_document_reference [added paragraph to section 3.2.3].
62	STAMP	technical	closed	2.0	Document_usage_constraint for document_file (see Issue #26)	What does it mean to "constraint the use of a file" ?	See section 10.3
37	STAMP	technical	closed	2.0	Use of product_definition.id for parts	The May 27 version of the RPG dropped both the pre-processor and post-processor recommendations for PRODUCT_DEFINITION.id (for Parts).	Previous use of the id attribute on product_definition had sometimes 'overloaded' the unique identifier with informal life cycle or organization information. This is not generally recommended for the PDM schema - this attribute should contain a unique identifier for the part view definition - no additional semantics are associated with this attribute in the PDM schema. Clarify section 1.1.1.4 [added text for pre- and post- processor recommendations].
40	STAMP	technical	closed	2.0	Product_definition_formation_with_specified_source for document (see Issue #39)	does PDFWSS.make_or_buy make sense for a document?	no - Clarify section 1.1.1.3 [added specific text excluding use in 'document as product']
52	STAMP	technical	closed	2.0	Use of document_file.name (see Issue #51)	Should the OS file name be specified inn DOCUMENT_FILE.name? DOCUMENT_FILE should be treated as external reference. The file name itself should be specified in "source property"	document source property (external_identification_assignment) should be used to contain source (path) information suitable to use for accessing a document or file. (see Issue solution #51).
53	STAMP	technical	closed	2.0	document_file multiple inheritance (see Issue #8)	document and characterized_object ... actually both define "name" and "description"	correct section 7.1.1.1 [added text to clarify]
55	STAMP	technical	closed	2.0	Use of document_representation_type	A question... p. 37 of RPG shows a P21 example of such an instantiation. However, it does not show DOCUMENT_REPRESENTATION_TYPE. Does anyone know why? Section 3.1.1.2 shows that when representing files with DOCUMENT_FILE you need to instantiate a DOCUMENT_REPRESENTATION_TYPE	incomplete example - clarify example 10 [added instance to example file fragment]
64	STAMP	technical	closed	2.0	multiple roles for one approval	How do we determine which APPROVAL_ROLE applies if two or more roles are assigned to the same APPROVAL?	the approval_role applies to the assignment of an approval_person_organization to the approval - multiple person_organizations may be assigned to a single approval in different roles ... (see solution Issue #13)
17	STEPwise	technical	closed	2.0	'raw material, 'tool for category'	see issue #3	add hint that says that PRPC will be revisted in the context of material. Is 'tool' in the scope of the UG? (see solution Issue #35)
16	STEPwise	major tech	closed	2.0	prefixing organization info is not good	Prefixing an organization identification to a product id is not a good recommended practice for uniquely identifying a part. Processors need to be able to evaluate more than one string (in this case product.id) to establish uniqueness of a part	The associated organization entity with the role of 'id owner' can be used as a uniqueness parameter if the product.id attribute is not unique within the domain of the business arrangement the exchange is to satisfy. Clarify section 1.1.1.1. [added text to clarify]
20	STEPwise	editorial	closed	2.0	revision control of external files	is the intention to say: use the document as product approach if managed revision control for files is desired	clarify to say 'yes' (see solution Issue #48)
25	STEPwise	major tech	closed	2.0	document_equivalence: assigned to the product (part identification) instead of the product_definition (part view definition)?		recommend assignment from product_definition_formation to indicate a particular document version to the view definition (product_definition) within the part master - clarify section 10.1.1.1 [added text to clarify]

issue number	author	classification	status	affected release	abstract	description	proposed solution
23	STEPwise	technical	closed	2.0	object_role: mandatory and informative	what are the semantics of these values, is this an exclusive list?	[see resolution Issue #58]
24	STEPwise	major tech	closed	2.0	Question the need to relate files directly to product data where the product data does not impose some level of configuration control over that file (see Issue #22)	In many PDM systems files are only attached / related directly to managed items that impose some level of configuration control. In these cases a PDWAD could just as well satisfy that file to managed item relationship.	[see resolution Issue #22]
26	STEPwise	major tech	closed	2.0	questioning constrained file or document reference	Constraining the usage of a file seems again to be treating a file as a document. . . If this is the intent, a product (document) instance needs to be created and the file(s) related to it through a PDWAD, then related to other documents or a part.	Is it reasonable to say you manage a document - say a description of fact A - that is physically instantiated as part of a file, e.g. a handbook describing facts A-Z? Thus you constrain the use of the physical file, that for some other reasons combines multiple aspects that are not - and should not be - represented in the managed document. Assignment of constraints to files and managed documents is currently supported. Requirement is outside the scope of the PDM Schema. [Proposed structure is not supported in the PDM Schema EXPRESS. A module might extend the PDM Schema to meet this requirement.]
27	STEPwise	technical	closed	2.0	organization.name	Disagree with the statement that the name attribute of the organization entity must contain a short identifier or acronym for the organization. The name attribute should be the name of the organization. If a short identifier or acronym needs to be captu	agree - reword to indicate the name attribute should contain the name of the organization - clarify section 13.1.1. [text added]
28	STEPwise	technical	closed	2.0	correspondance between approval and org/persons	There is a more comprehensive approach that could be use to satisfy when an approval event is a cycle, which requires multiple people to concur on possibly differing dates and /or times. The approach is to instantiate an approval instance for each person	separate doc available from STEPwise team [see solution Issue #13]
72	ProSTEP	technical	closed	2.0	use of 'ANY' vs. '/ANY'	special string 'ANY' for the version letter to indicate any version is acceptable - '/ANY' more consistent with other special string '/NULL'	while '/ANY' may be consistent with '/NULL' in the use of the slash, the string 'ANY' was originally used in pilot/production use to indicate the semantic that any existing version of a component is valid for the parent assembly - there is a legacy of pilot data using the 'ANY' string - is there real value added to change to '/ANY' just for consistency? Darm899: discuss with EF proposed change to 'ANY', otherwise reject issue (both '/ANY' and 'ANY' will be accepted). Talk with EF evidenced that it needs '/ANY'. AP 203 RPG recommends 'ANY'. Agreed to recommend '/ANY', as 'ANY' might be used as a valid version identifier [modified section 1.1.1.2]
22	STEPwise	major tech	closed	2.0	Question the general use of applied_document_reference to assign documents to product data	Do not see the need to ONLY use document_reference to associate document to product data. In many cases using a product_definition_formation_relationship or product_definition_relationship is much more straight forward and direct. This is done in AP 232	Glen Zlioko: support as an optional method, recommended when the assignment of properties to the relationship/association is needed ????? [This was not identified as a common requirement, therefore considered outside the scope of the PDM Schema. A modular extension to the PDM Schema may meet this requirement.]
65	BaE	technical	closed	3.0	Supplied part identification	a supplied part is ... mentioned in the usage guide, but only in the context of other recommendations, i.e. don't use an alias for a supplied part. However, there is no description ... as to how to represent it.	[see section 4.6.4]
30	PDMI2	technical	closed	3.0	recommendation for values for document properties	existing values are not useful	use values recommended by E-Leiter Kreis (separate doc available) Values in the PDM schema usage guide are consistent with E-Leiter Kreis document (see section 9). Additional values may be made available as part of a companion document.

issue number	author	classification	status	affected release	abstract	description	proposed solution
50	STAMP	technical	closed	3.0	Use of document_type (see Issue #21)	What are the recommended values for DOCUMENT_TYPE.product_data_type when assigned to DOCUMENT_FILE entity?	(see Issue #49)
68	STAMP	technical	closed	3.0	Part/document/file properties	It is our understanding that all properties relating to parts or documents or files are specified using the DESCRIPTIVE_REPRESENTATION_ITEM construct applied on the object (part/doc/file) using PROPERTY_DEFINITION.	[see sections 3 and 9]
21	STEPwise	editorial	closed	3.0	Need to recommend what to instantiate the required entity document_type with	no recomendation in that section	add cross-reference to section doc properties (see Issue #49)
54	STAMP	technical	closed	3.0	applied_external_identification_assignment use (see Issue #52)	thought that an external reference (in the context of an exchange of data files) would indicate to the recipient that the file is not included in the exchange package, but rather may be found in a certain location for download...the file path information should also be accompanied by host machine information	[The usage guide release 4.0 recommends separating the storage node identification from the path information. See section 9.7]
12	PDM Editor	technical	closed	4.0	approval_status.name	are values besides 'not yet approved' 'approved' 'dissapproved' and 'withdrawn' allowed?	clarify - added paragraph in general section explaining how to handle string values, i.e., use values where applicable. Remove restrictions on approval status values.
31	PDMI2	minor tech	closed	4.0	recommended values for approval status (see issue #12)		[see resolution Issue #12]
49	BaE	technical	closed	4.0	Use of document_type (see Issue #47)	There is no recommended value for document_type.product_data_type. In fact the usage guide indicates in the part21 examples to use 'unspecified'. We are going to use the MIME type! E.g. doc, pdf, etc. However, ... should use document properties resulting in descriptive_representation_item('data format', 'word')?	The value of document_type.product_data_type is discussed in more detail in the future section on document properties. For an external file, document_type should in general be used to classify the kind of product data that is contained in the file e.g. 'FE data', 'NC data', 'process plan', etc. Mime type information seems more like the document format property (see section 9.4) [for external file type, added section 9.8.1; for document as product type added section 6].
66	BaE	technical	closed	4.0	Alias identification	The UG mentions aliases, again in the context of other recommendations, but provides no guidance on how to use applied_assignment_identification as an alias.	[added section 12]
6	PDM Editor	editorial	closed	4.0	product_related_product_category and document classification	the UG could be read to say that the category allows document classification	[added section 6]
73	PDM UG team	technical	closed	4.0	Don't require assignment of role of a person within an organization		[Wording requiring use of name attribute removed from section 13.1.2.2]
44	STAMP	technical	closed	4.0	Document_relationships (peer-to-peer, parent-child, substitutes, alternates)	So in PDM Schema, how are document relationships specified (or instantiated)?	[added section 11]

PDM SCHEMA USAGE GUIDE – ISSUES LOG

issue number	author	classification	status	affected release	abstract	description	proposed solution
69	STAMP	technical	closed	4.0	file to file relationships	It is our understanding that File-to-File relationships for Copy, Sequence, Derivation, Reference, and Substitution are done using DOCUMENT_RELATIONSHIP entity. If this is not accurate, we need to know.	[added section 11]
70	STAMP	technical	closed	4.0	Notes - how are notes represented in PDM schema?	are the NOTEs documents (which specs are) or are NOTEs just a collection of descriptive text applied to a particular part, document, or any product data?	both eurofighter and stamp pilot projects have mapped 'descriptive text' notes/remarks to descriptive rep items related via property definition - this captures the actual text in the STEP file. This is as opposed to representing notes as a 'document' or a 'file', where the actual text is not captured but referred to. add explanation of the mapping of text notes/remarks as property?? [added section 9.5 for document notation and paragraph on section 3.1.3 for part remarks]
47	STAMP	technical	closed	4.0	Document categorizations (see Issue #6)	The May 27 version of PDM schema RPG does not mention anything on Document categorization. Hence we propose to use categorization similar to what is in AP203.	[added section 6].
18	STEPwise	technical	closed	4.0	product_category_relationships should not be restricted to exist only between PRPC	Question the recommended practice of having a processor just handling PCR between two PRPC and not between PRPC and PC, or PC and PC.	Describe this capabilty for parts as a more generalized part type classification capability in a future section (Part Classification). This section should include product_category (as itself, not subtype prpc) and allow for user defined name attribute values. [added section 2 and made postprocessor recommendation for product_category in section 2.1.1.1]
19	STEPwise	major tech	closed	4.0	Need to clarify the statement that "Document as Product" PD's are used to build document structure	Does this mean that there is only a mechanism to specify structure for representations, but no way to specify structure for actual documents?	clarify the semantics of structure on product_definition and product_definition_formation level in a future section Document Structure and Relationships. How do they interact?? [added section 11]
74	GOSET	technical	closed		Named unit in the context of quantified_assembly_component_usage.quantity is ambiguous		In the specific context of quantified_assembly_component_usage, it is not ambiguous, as what is meant is the number of components.
80	EF	technical	closed	4.1	date only for approvals	should describe the use of date only rather than just date_time for approval_date_time	add description of the use of these constructs as indicated - modifications made to section 13.2
99	GOSET	technical	closed	4.1 24	make_from_usage_option inconsistent example	section 4.6.3 example 24 gives misleading information: the description text for #16 is 'make solid box from two solid cubes', but the structures really represent 'make two solid boxes from one solid cube" …	And the recommendation should be consistent with the example in part 41 (i.e. required use of context_dependent_unit).

issue number	author	classification	status	affected release	abstract	description	proposed solution
90	GOSET	technical	closed	4.1	instantiation of quantified_assembly_co mponent_usage (context_dependent_uni t)	Reading the PDM schema usage guide I have found it a bit "shy" about the recommendations for the instanciation of quantified_assembly_component_usage . Let's consider we want to describe an assembly where 4 occurrences of a given bolt are used. What kind of instance is expected in the attribute quantified_assembly_component_usage .quantity? More precisely what is the expected unit of the measure_with_unit that is referred to by this attribute? Looking at the example you provide in your guide (e.g. in clause 3.1.2.1), it seems that you only require a named_unit with all dimensional_exponents equal to 0. This is for me unsufficient as many units may have such dimensional exponents (e.g. a currency unit). In addition, this coding is not consistent with the example provided by ISO/DIS 10303-41 in annex F3 (see - example 3) and with the example provided in the same part in the definition of the entity context_dependent_unit (clause 21.4.5)	usage guide should provide a more precise recommendation about the way to define the unit for such a case. As a result of the WG12 workshop, the examples for context_dependent_unit.name provided in ISO 10303-41 clause 21.4.5 and annex F.3 will be made consistent with ISO 10303-303 ATC 6.16 and ETC cm15.2. Therefore, in these examples, the existing words will be replaced by 'parts'. Introduce a new chapter to describe the use of measures and units. This chapter will describe the use of si_units as well as count/each/part - additions made in new section 16.
122	JSTEP	technical	closed	4.1	Diagram 2 incorrect attribute	aplication_protocol_definition.application s is not a SET attribute as indicated in the diagram.	
123	JSTEP	editorial	closed	4.1	name attribute incorrect type	type: label = string	
125	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 5	see issue 124	
126	JSTEP	technical	closed	4.1	application protocol year in example 5	The application_protocol_year sould be 1998 (not 1999). #80=APPLICATION_PROTOCOL_DEFI NITION('version 1.1',pdm_schema',1998);	The correct year value to use is the year that version 1.1 of the pdm schema was released - 1998. This value should be used consistently throughout all examples - modifications made accordingly.
127	JSTEP	editorial	closed	4.1	representation_context	same as issue 87	
128	JSTEP	editorial	closed	4.1	missing OPTIONAL	description attribute should be labeled OPTIONAL	
129	JSTEP	editorial	closed	4.1	DERIVED attributes?	should DERIVED attributes be listed in the entity table? Need to be consistent.	Add text in the general information section to explain that not all derived attributes are described in the entity tables - in general derived attributes are not described in the entity tables (in accordance with the part 21 specification where they do not appear in the exchange file) - only when special attention is required will such an attribute appear - modifications made in general information.
130	JSTEP	editorial	closed	4.1	shape_definition_repres entation.definition	Strictly speaking, the definition attributes should points PROPERTY_DEFINITION, not PROPERTY_DEFINITION_SHAPE.	This issue is based on a misunderstanding of the diagrams - they represent an instance model derived from the EXPRESS schema and are not EXPRESS-G - reject

issue number	author	classification	status	affected release	abstract	description	proposed solution
132	JSTEP	editorial	closed	4.1	ref_direction	The attribute name 'reference_direction' is not correct. It should be only 'ref_direction'.	
134	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 11	see issue 124	
135	JSTEP	technical	closed	4.1	complex shape_representation_r elationship	In spite of the title is shape_representation_relationship, the entity instatiation attributes are based on complex entities of 'shape_repersentation_relationship', 'representation_relationship_with_ transformation' and 'representation_relationship'. Then in the instantiation table, it needs some considerations ??	This issue is editorial - modifications made to section 3.3.1
136	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 12	see issue 124	
137	JSTEP	technical	closed	4.1	Diagrams 14, 15 misleading	Strictly speaking, the local_origin attribute does not come from CARTESIAN_ TRANSFORMATION_OPERATOR_3D, but its supertype CARTESIAN_TRANSFORMATION_ OPERATOR.	same as issue 130 - reject
138	JSTEP	editorial	closed	4.1	scale attribute is optional - dotted line	Diagram 15 illustrates OPTIONAL attribute but doesn't show dotted line.	
140	JSTEP	editorial	closed	4.1	no underscore in axis attribute names	axis_1 -> axis1, etc.	
141	JSTEP	technical	closed	4.1	Example 14, #1040 -> #1000, #1070 -> #1040	#1040=CARTESIAN_TRANSFORMATI ON_OPERATOR_3D('',' ',#1020,#1030 ,#1010,$, #1000); #1050=REPRESENTATION_RELATION SHIP_WITH_TRANSFORMATION('',' ,# 800,#900, #1040);	check the validity of these constructs and change as appropriate - modifications made to section 3.4.3
144	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 25 and Diagram 26	see issue 124	
145	JSTEP	editorial	closed	4.1	example 21 error, #730 > #800, #750 -> #820, #790 -> #870, #760 -> #830	#810=SHAPE_DEFINITION_REPRESE NTATION(#800,#820); #820=SHAPE_REPRESENTATION('sa2 , #870, #830);	check the validity of these constructs and change as appropriate - modifications made to section 4.4.3
131	JSTEP	editorial	closed	4.1	DERIVED attributes?	see issue 129	same as issue 129 - reject
142	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 24	see issue 124	
143	JSTEP	editorial	closed	4.1	representation_map.ma pped_representation	value is entity = representation, not mapped_representation	
146	JSTEP	editorial	closed	4.1	two relating attributes?	same as issue 100	
147	JSTEP	editorial	closed	4.1	two relating attributes?	see issue 146	
148	JSTEP	editorial	closed	4.1	consistent use of SELF?	Other table except this one are not using 'SELF' notation. It seems better they are done in the same way.	This use of SELF is a special case due to the multiple inheritance of attributes having the same name - the SELF distinguishes the attributes defined in document from those with the same name defined in characterized_object - reject

issue number	author	classification	status	affected release	abstract	description	proposed solution
149	JSTEP	editorial	closed	4.1	Diagram 37 incorrect	document_file.representation_types -> document_file.represented_document, applied_identification_assignment.assigned_items -> applied_identification_assignment.items	
150	JSTEP	editorial	closed	4.1	Figure 10 incorrect	MP-032 -> MP-03-2	
152	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 50	see issue 124	
153	JSTEP	editorial	closed	4.1	organization_role	person_and_organization -> organization attribute description is DERIVED, not OPTIONAL	
154	JSTEP	editorial	closed	4.1	person_and_organizatio n	attribute name :: type label = string attribute description :: type label = string, DERIVED, not OPTIONAL	
156	JSTEP	technical	closed	4.1	Example 62	This file contains '#0' statement, which is generally not recommended.	This is an non-controversial issue - modifications made to section 14.2.1
157	JSTEP	technical	closed	4.1	Example 63	see issue 156	This is an non-controversial issue - modifications made to section 14.2.2
155	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 58	see issue 124	
159	JSTEP	editorial	closed	4.1	incorrect # numbers in Diagram 69	see issue 124	
95	PDMI 2	technical	closed	4.1	quantifed instance with n=1	add recommendation that for a quantity of one a simple instance of NAUO should be used.	related to issue 90 and will be addressed by the new chapter that describes the use of si_units - additions made to section 16.
96	PDMI 2	technical	closed	4.1	units for quantified instances	add recommendation that - where applicable - pre-defined units shall be used to define quantities for component instances	addressed by the new chapter that describes the use of si_units- additions made to section 16.
101	PDMI 2	technical	closed	4.1	document type classification table	the table 2 (basic document classifcation hierarchy) should be extended. Include geometry in 2nd level. Values for 3rd level?	based on reviewers' feedback, team decided to remove table.
79	PDMI 2	technical	closed	4.1	applied_date_assignme nt for approval_person_org	applied_date_assignment (instead of applied_date_time_assignment) should as well be allowed, see diagram 56	add description of the use of these constructs as indicated - modifications made to section 13.2
108	PDM-if	technical	closed	4.1	application_context	clarify the use of this entity relative to product_definition_context and product_context (use one for all product_context that has not value for attribute application, use others for the product_definition_context that have values for application as currently described).	Modifications made to sections 1.1.2 and 5.1.2
84	PDM-if	technical	closed	4.1	example 7 incorrect	entity #5020 in example 7 page 36 has value 'mass property' in wrong attribute - in the description rather than the name.	fix example
85	PDM-if	editorial	closed	4.1	Diagram 6 incomplete	Diagram 6 doesn't include representation_context in the picture!	

issue number	author	classification	status	affected release	abstract	description	proposed solution
86	PDM-if	editorial	closed	4.1	representation for parts	entity representation talks about *document* properties – but this is the part property section!	
87	PDM-if	editorial	closed	4.1	representation_context	entity should be representation_context, but the entity table lists geometric_representation_context	
89	PDM-if	technical	closed	4.1	need example for general property	example of the use of general property would be useful	change the part21 example to have two uses of an applied property (e.g. mass property) both collected together using a general property definition - modifications made to section 3.1.2.3
91	PDM-if	technical	closed	4.1	clarify CAD file assignment to parts	clarify the recommendation in section 3.2.3 regarding CAD files that are the constituents of a managed document.	Add to section 3.2.3 something like " This reference should be applied directly from the external file if it exists alone as an unmanaged document; if the external file exists as a constituent file of a managed document, the reference should be applied from the managed document via document_product_equivalence". modifications made to section 3.2.3
92	PDM-if	technical	closed	4.1	example 15 error	entity #420 is not well formed, required description attribute is missing	fix example
					assembly definition' as a	discussions have led to a general understanding of what is meant by 'part definition' and 'assembly definition' for the context of a product_definition: - 'assembly definition' : this view of the part version has assembly structure provided. A processor may expect to have assembly-component relationships with this as the parent. - 'part definition' : this view of the part version does not have assembly structure provided. Given this, a view is either one or the other, does it make sense to be both?? Examples 15, 16, 17, 18, 19, 20 and 21 and the 'pre-processor recommendations' text of section 4.1.1.2 are inconsistent with the above understanding.	Change the examples 15, 16, 17, 18, 19, 20 and 21 and the 'pre-processor recommendations' text of section 4.1.1.2 to be consistent with the above understanding - clarifications made to text in section 1.1.2.4; the 'pre-processor recommendations' text of section 4.1.1.2 and the text in section 1.1.2.4 to make clear that 'assembly definition' is an optional addition (for AP214) that may be instantiated to indicate that the product_definition has component structure provided beneath it within the STEP exchange file. This is not required - the same information may be obtained by simply traversing the product structure provided in the file.
94	PDM-if	technical	closed	4.1	product_definition_cont ext name	Note whether a part is in general an assembly or detail from the point of view of the 'id owner' is represented by the product_related_product_category. A part may be an assembly, but an exchange may contain only the 'part definition' view of the part ...	
97	PDM-if	editorial	closed	4.1	organization of sections 4.4, 4.5, and 4.6	the logic of sections seems to indicate the sections should be numbered 4.4, 4.4.1, and 4.4.2 rather than 4.4, 4.5, and 4.5.2 respectively	change section numbering
98	PDM-if	editorial	closed	4.1	example 21 error	example values for product_related_product_category instances #40 and #50 have capital letter for 'Part' and 'Assembly' - recommendation in section 1.1.3 indicate 'part' and 'assembly'	fix example

issue number	author	classification	status	affected release	abstract	description	proposed solution
100	PDM-if	editorial	closed	4.1	two relating attributes?	section 4.6.4.1 lists two attributes with the same name : relating_product_definition	change the text so that the relating_xxx references the internal(original) part number and the related_xxx references the vendor/supplied(alias) part number.
161	PDM-if	technical	closed	4.1	version hierarchy as well as iteration?	pdm systems manage iterations as well as versions (this is modelled in the OMG PDM enablers) - need a version hierarchy relationship along with version sequence to provide complete version history	accept, modifications/additions to section 4.5.5 - should add additional examples from SAVE project input to remove the known issue listed about the use of both sequence and hierarchy together??
93	PDM-if	technical	closed	4.1	organization type	Propose to introduce recommended values for the attribute organization.desription to represent the "organization type". The following values should be used if appropriate, otherwise make up your own: - Company - Department	This is an non-controversial issue - proposal to add an open ended 'permissive' list to the affected section that contains the two values 'company' and 'department' - modifications made to section 13.1.1
76	PDM-if	editorial	closed	4.1	spelling/typo	oorganization -> organization	
102	PDM-if	editorial	closed	4.1	clarify document source property	clarify use of document source property and document_file id	clarify the mapping of the file name for external files - document_file.id is the unique identifier (file name) of the file. The source property may in addition be assigned to a document file to provide access path information - the access path information is composed of two parts - the "storage node identification" and the "path information" as described in 9.7. the current usage guide is not incorrect, but maybe could be clarified in text of section 9.7. - clarification made to section 9.7
160	PDM-if	technical	closed	4.1	'work definition' mandatory?	in a work order, the instance of directed_action representing the top-level 'work definition' for a work order is required to have the name 'work definition' - since the directed_action itself represents the work definition, this additional string may not be necessary. This is particularly true in the case when no decomposition of the work definition is done, and the name may be useful to characterize the work better than 'work definition'	accept, change 'shall' to 'may' - modifications made to section 15.2.1.2
162	PDM-if	technical	closed	4.1	derived id attributes	the derived id attribute for both directed_action and executed_action are not documented	accept, modifications/additions to section 15.2
82	PDM-if	technical	closed	4.1	clarification of approval cycle use of the approval.level attribute	the use of the attribute approval.level is also an important distinction between the requirements for a single approval (and thus a single approval.level) with multiple person/organizations and that for multiple approvals (and thus multiple approval.levels) in a hierarchy	clarify the explanation in section 13.2.2 of when one should use which of the two ways described for approval cycles and multiple sign-off scenarios - see issue 81 resolution

PDM SCHEMA USAGE GUIDE – ISSUES LOG

issue number	author	classification	status	affected release	abstract	description	proposed solution
104	PDM-if	editorial	closed	4.1	include ARMish diagrams	the PDM-if has introduced ARMish instance diagrams as a way to clearly illustrate the user requirements and semantics intended for preprocessing test data . These diagrams use the ARMish terms provided by the PDM schema usage guide (generally taken from the table of contents). These diagrams would be a good addition to the usage guide itself.	include ARMish diagrams where appropriate/available in the usage guide. Such a diagram should generally be placed just before the standard sub-section labeled "The Instance Model: EXPRESS entities and attributes". Although these diagrams were developed at the specific request of and considered quite helpful by the PDM-if, this issue was rejected.
105	PDM-if	technical	closed	4.1	measure and units	at a number of occasions measure with units are to be used (quanitifed instances of components, part properties) the UG does not give general guidance on measure and units	add corresponding section - additions made in section 16.
106	PDM-if	editorial	closed	4.1	add an index	would be good to have an index that identifies the main "ARMish" requirements and all the AIM entities in the document.	mark for index the current TOC entries (for the ARMish requirements) and the 4th level sub-headings (which are the AIM entities)
81	ProSTEP	technical	closed	4.1	mapping for approval.level	section 13.2.1.1 recommends no standard mapping for attribute approval.level - but it is commonly agreed to be the approval purpose/type in terms of its usage.	change preprocessor and postprocessor recommendations in section 13.2.1.1 to reflect the common interpretation of the level attribute - see issue 78 resolution
					document source	identification_role.name should be 'document source'. The text for the description should not be such strong. It is general UG policy not to use the word 'must' - in fact description should be optional. (only required for 232 compatibility?). Remark: if 'must' is acceptable wording in general it should be introduced at a number of other places where it is applicable as well.	
103	ProSTEP	technical	closed	4.1	property	same as issue 84	This issue is the same as issue 102 - clarification made to section 9.7
88	ProSTEP	technical	closed	4.1	example 7 incorrect	same as issue 84	fix example

issue number	author	classification	status	affected release	abstract	description	proposed solution
78	ProSTEP	technical	closed	4.1	clarification of interaction between approvals and the things approved, particularly when the things approved have context information (e.g. product_definition)	We might have to add some explaination how approval, life cycle stage, approval type and status of approval play with each other. The question I ask myself with the examples from p41 below: - prelim design completed - released for production (i.e. final design) Would these - if we talk about the relations approvals of product_definitions (part views) not be related to different product_definitions, that establish cooresponding context-dependent views, anyway? Maybe we need to clarify that a little further?	illustrate the use of the attributes approval.level and approval.status. Briefly discuss the issue of the business decision to make a new product/version/view definition versus using approval levels to describe/define the "states and gates" of a product through the life cycle - modifications made to section 13.2
75	STAMP/STEP wise	major technical	closed	4.1	where to capture item/find number	There is a conflict with using the same attribut	It will be documented as a known issue. AP 214 uses NAUO.name attribute to differentiate single instances from selected instances. [added as known issue in section 4.8.2, submitted AP 214 DIS ballot comment] This ballot comment was accepted, and so we can possibly close this issue, remove the known issues section, and add explanation text regarding find number to section 4.1.1.2. AP 232 does not capture the find number in a NAUO but in a product_definition_relationship entity between the product_definition of the part and the product_definition of a drawing document. The top assembly in a parts list does not have a NAUO with its Find number, so to capture the top assembly's Find number a relationship between a part and a drawing is needed also. This general mapping between a part and a drawing document can satisfy capturing both assembly and component Find numbers. Provide a comprehensive example of a typical industry data exchange package, including a part list/bill of material, a drawing, item find numbers, reference designators, and instance identifiers (see issue 107) - modifications/additions made to section 4.1.2
76	BAe	major technical	open	?	how to exchange net chan	Current net change approach in 203 RPG doe	This issue is currently out of scope for the usage guide - it will be left **open** against release 4.1
133	JSTEP	editorial	open	future	incorrect # numbers in Diagram 9	see issue 124	unable to edit diagram 9 (Identification and Representation of Portions of Part Shape Instance Diagram)
139	JSTEP	editorial	open	future	Diagrams 16, 18 misleading	Diagrams illustrate OPTIONAL attributes, but don't show dotted line.	diagram 18 fixed; unable to edit diagram 16 (Explicit Representation of a Translation with Rotation Instance Diagram)
151	JSTEP	editorial	open	future	incorrect # numbers in Diagram 45, Diagram 46, Diagram 47, Diagram 48	see issue 124	unable to edit diagrams
158	JSTEP	editorial	open	future	incorrect # numbers in Diagram 66	see issue 124	unable to edit diagram 66 (General validity period effectivity Instance Diagram)
124	JSTEP	editorial	open	future	incorrect # numbers in Diagram 4	For instance entities such as APPLICATION_CONTEXT, the #1 mark should be deleted or corrected in accordance with its sample files. In this diagram, there are duplicated #-numbers like #1 for APPLICATION_CONTEXT, PRODUCT_DEFINITION_CONTEXT and also PRODUCT_RELATED_PRODUCT_CATEGORY.	unable to edit diagram 4 (Complete Part Master with Context and Type Classification Instance Diagram)

issue number	author	classification	status	affected release	abstract	description	proposed solution
83	PDMI 2	technical	open	future	date_time_role	there is no description on date_time_role in the context of applied_date_time_assignment	This issue needs clarification - *ProSTEP to provide additional input to clarify what the issue means.*
107	PDM-if	technical	open	future	comprehensive TDP example	annex should include a TDP including parts list, assembly drawing, and ECO that all go together and relationship to the data structures … this should clarify the use of find number, reference designator, and instance identifier.	
10	PDM Editor	major technical	open	future	approval.scope	scopes for approval not covered in UG	the assignment of an organization to an approval in the role of "scope" of the approval is done in AP214 - this requirement is not possible in the PDM schema - organization_item select type does not include approval [added open issue text] Add to the list of select types extensions under consideration. Usage guide will ensure that this capability is allowed but not mandatory.
67	STAMP	technical	open	future	Security_classification person/organization and date/time	It doesn't make sense to not have PAO and Date/Times associated with Security-classifications.	allow person/organization and date/time to be assigned to security classification if desired - (not mandatory) - but security_classification is not in the required select types in the pdm schema for this. Currently not supported in the PDM schema AIM. It requires extending select types. [AP 214 DIS ballot comment has been submitted].
109	PDM-if	technical	open	future	entity document	inverse attribute representation_type -> representation_types	
110	PDM-if	technical	open	future	document_file as select value	should add document_file to select types approval_item and date/time_item	
111	PDM-if	technical	open	future	product_definition_form ation as select value	should add product_definition_formation to select type identification_item	
112	PDM-if	technical	open	future	property_definition as select value	should add product_definition_formation to select type document_reference_item	
113	PDM-if	technical	open	future	EXPRESS rules in AP214	There are 152 global rules in AP214 cc6, but only 4 global rules in PDM schema. We need to check the correspondence of these AP214 rules with the PDM schema usage guide	
114	EF	technical	open	future	versioned_action_reque st as select value	should add versioned_action_request to select type document_reference_item	
115	ProSTEP	technical	open	future	organizational_project as select value	should add organizational_project to select type date/time_item	
116	PDM-if	technical	open	future	product as select value	should add product to select type identification_item	
117	PDM-if	technical	open	future	action_directive as select value	should add action_directive to select type identification_item	
118	EF	technical	open	future	versioned_action_reque st as select value	should add versioned_action_request to select type name_attribute_select	
119	PDM-if	technical	open	future	project as select value	should add project to select type approval_item, person/org_item, and date/time_item.	
120	PDM-if	technical	open	future	approval as select value	should add approval to select type person/org_item.	
121	PDM-if	technical	open	future	action as select value	should add action to select type organization_project_item.	