Recommended Practices
for
External (Element) References

Release 3.1

January 20, 2014

Contacts	
Jochen Boy	Phil Rosché
ProSTEP iViP Association	PDES, Inc.
Dolivostraße 11	5300 International Blvd.
64293 Darmstadt / Germany	North Charleston, SC 29418 USA
jochen.boy@prostep.com	phil.rosche@scra.org

Table of Contents

List of Figures

List of Tables

Document History

This document replaces the following CAx-IF Recommended Practices:

- PDM Schema Usage Guide; Release 4.3; published January 2002
- Recommended Practices for External References; Release 2.1; published January 19, 2005
- Technical Discussions on External Element References (EER); Revision 3.2; published October 12, 2012

The current document covers the scope of the preceding ones as far as relevant for the scope of this document, and adds new and updated concepts.

Revision	Date	Change
3.0	2012-11-27	Major revision based on preceding documents
3.1	2014-01-20	Inclusion of contents from the PDM Schema Usage Guide as sections 5 (Fundamental Concepts) and 6.3 (Document and File Properties)

1 Introduction

In current business processes, the digital definition and description of the products under development is the most important artifact to handle. The increasing complexity of these products, and amount of data per product due to the advancements in model-based 3D engineering, have made it impossible to manage all data for one product in a single file.

STEP therefore offers a mechanism that allows for a distributed definition of a product by creating External References from a source or master file to one or more target files (which may in turn reference additional files), which contribute to the complete product definition by describing integral aspects of the definition (e.g. component geometry) or by indicating additional reference information (e.g. modeling standards used in the creation of the model, material specifications, etc.). In the case of assemblies, especially large ones, a common usage scenario for External References is to break down the assembly structure into several files, each defining one level of the hierarchy with iterative references. This is referred to as Nested External References

This document describes the recommended practices for the creation of External References, i.e. references to information stored at an external location. The reference target can be another digital file, which can be basically be of any file format (STEP, native CAD, Office format, …) and may be stored on a file system, in a database, or on the internet; or it can be a physical document (printout) with a specified storage location – the reference mechanism is very flexible in this respect.

The External Reference mechanism has been applied successfully in business processes for many years now, based on AP203 and AP214. During the creation of AP242, new use cases emerged that required it to not only reference an external file, but a specific element within the file. This new capability is hence referred to as External Element References or EER for short. This document describes the generic mechanism for EER, while the specific use of EER for a certain use case will be described in the corresponding domain-specific recommended practices.

2 Scope

The following are within the scope of this document:

- The definition of External References from a STEP AP203, AP214 or AP242 Part 21 file to an external document
- The definition of Nested External References for assemblies
- The provision for document management information
- The provision for document format properties
- The definition of generic External Element References from a STEP AP242 Part 21 file to an external document
- The reference to information that is stored in a manner different from digital files

The following are out of scope for this document:

- The definition of domain or use-case specific External Element References
- The handling of inbound External Element References in file formats other than STEP
- The definition of External (Element) References in STEP Part 28 (XML) files.

3 Document Identification

For validation purposes, STEP processors shall state which Recommended Practice document and version have been used in the creation of the STEP file. This will not only indicate what information a consumer can expect to find in the file, but even more important where to find it in the file.

This shall be done by adding a pre-defined ID string to the `description` attribute of the `file_description` entity in the STEP file header, which is a list of strings. The ID string consists of four values delimitated by a triple dash ('---'). The values are:

```
Document Type---Document Name---Document Version---Publication Date
```

The string corresponding to this version of this document is:

`CAx-IF Rec.Pracs.---External (Element) References---3.1---2014-01-20`

It will appear in a STEP file as follows:

```
FILE_DESCRIPTION(('...','CAx-IF Rec.Pracs.---External (Element) References-
--3.1---201-01-20',),'2;1');
```

4 Terminology

Different terms are being used to indicate the direction of an external reference, or which end of the reference is being discussed. The following terms are used synonymously throughout the document:

In the file from where there is a reference to another file	In the file which contains elements that are being referenced from other files
outbound	inbound
referencing	referenced
source	target
start	end
link	anchor

Table 1: External Reference Terminology

There are two levels of External References:

- "**classic**" or "**basic**" External References relate to the scenario where either no assembly structure is involved, or it is defined completely within the source file, and references to external files are given.
- "**extended**" or "**nested**" External References relate to the scenario where the assembly structure is defined in a distributed manner across several files, typically with one level of hierarchy per file. This means that an entire structure of files needs to be navigated to get to a leaf-node component.

For the description of External Element References, three levels of complexity will be used to introduce the applied concepts:

1. **Level 0** – everything in one file. Though this does not require any external references, it used as a baseline for the subsequent levels.
2. **Level 1** – Basic External Element References; i.e. no assembly structure, or entire assembly structure in one file, and part (geometry) definitions in external files.
3. **Level 2** – Nested External Element References; i.e. assembly structure is split into several files as well.

5 Fundamental Concepts

5.1 External Part Shape

In many data exchange scenarios, the default approach is to store or exchange the part master data separated from the geometry of the part shape. This means that there is one file defining the part master data (there may be several in the case of nested assemblies), and one additional file for the externally defined geometry of each component part. These geometry files do not necessarily have to be STEP files as well; they can also be provided in a native CAD format.

In the part master STEP file, the externally defined geometry is identified as an external representation of the part shape that is related to the part master data.

5.1.1 Geometric Shape

Part geometry is identified 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 specialized to the subtype `product_definition_shape`.

The external part shape is represented by the entity `shape_representation`, which is a subtype of `representation` used in general part properties. `Shape_representations` model external models that are referenced in files associated to the part 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 External References, it is generally assumed that `shape_representations` are placeholders for externally defined geometry. In general it is not recommended to instantiate dedicated geometric elements as items of the `shape_representations` for part master data 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 EXPRESS entities and attributes used to support the requirements of independent property identification are illustrated in Figure 1.

Part 21 Example:

```
#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);
```

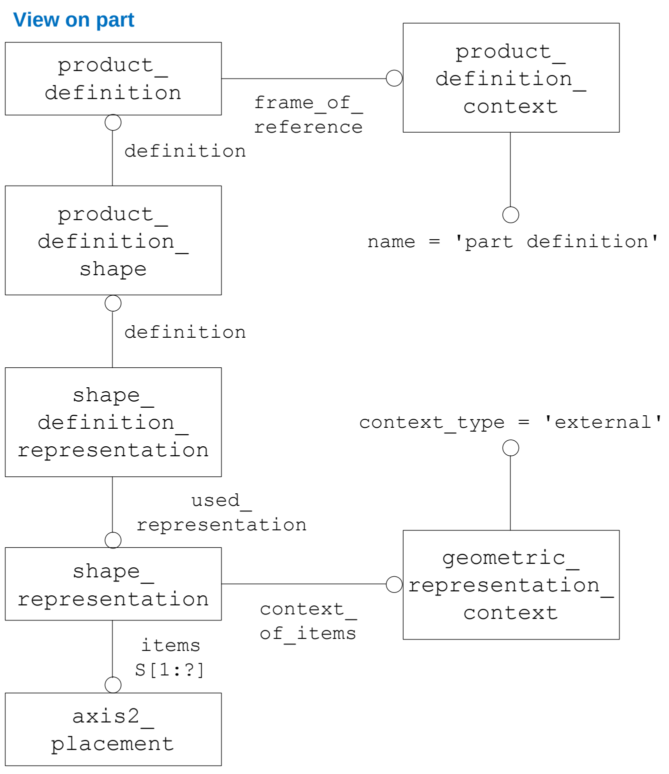

Figure 1: Assignment of Part Shape to the View of the Part

5.1.2 Relating Externally Defined Part Shape to an External File

Part geometry is identified in STEP as a representation of a property of a part definition. In the scope of this document, the geometry is identified as an external representation of the part shape that is related to the part master data - it is actually in externally referenced files. These files are typically generated by CAD systems, and contain the detailed geometric shape representation with dedicated geometric elements.

Specifics of external file identification are described in section 5.2. 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. 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 5.2.4). 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 5.5).

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 EXPRESS entities and attributes used to support the requirements of relating part shape to the external CAD file are illustrated in Figure 2.

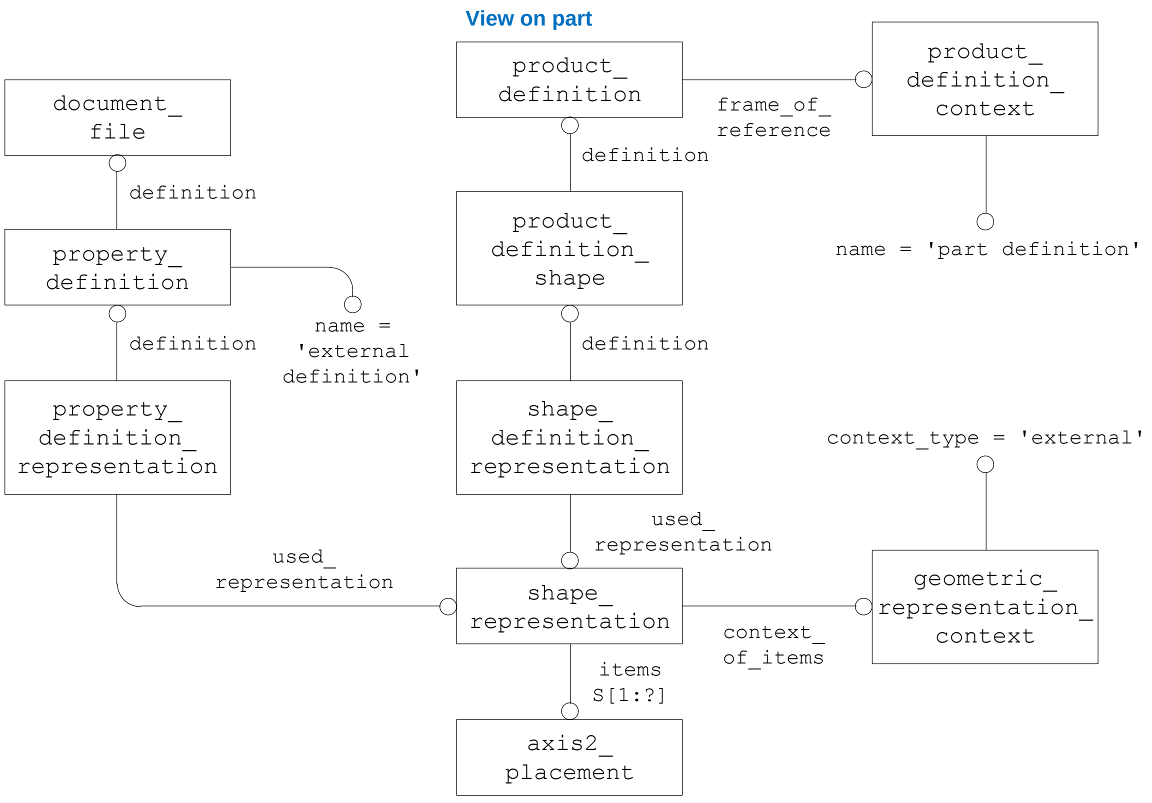

Figure 2: Part Shape Related to External File

5.2 External Files

External files in STEP represent a simple external reference to a named file. The external file may identify a digital file or a physical, 'hardcopy' 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 (see section 5.3 below).

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.

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 EXPRESS entities and attributes used to support the requirements of external file identification are illustrated in Figure 3 below.

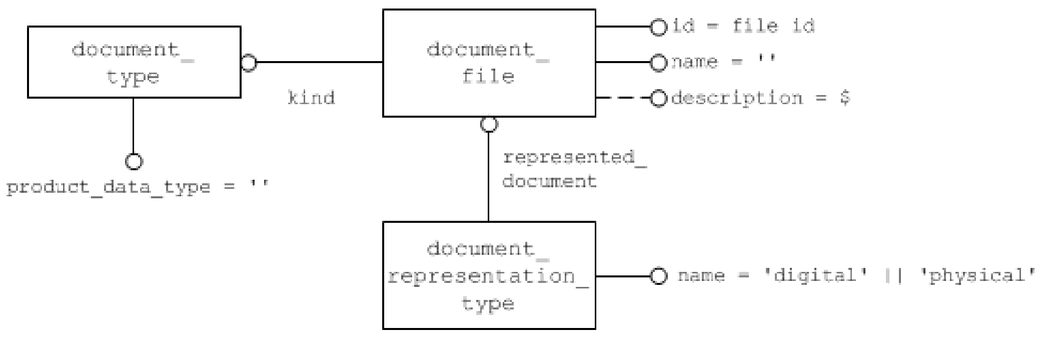

Figure 3: External File Identification

5.2.1 Document File Identification

5.2.1.1 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 super-types of this entity define local attributes `name` and `description`, the sub-type `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 super-type `characterized_object` should not be used for valid user data. These constraints specify that the value of the name attribute be an empty string: '' and the optional description attribute should not be populated: $.

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` super-type 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 super-type `document` is valid for user data. The `description` attribute inherited from `characterized_object` should not be populated ($).
- 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 `$`	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 inheritances of the attributes name and description. An example instance of document_file is illustrated in section 5.2.1.3.

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 5.2.1.2 below.

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.

5.2.1.2 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 a 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.

5.2.1.3 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.

Part 21 Example:

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

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 EXPRESS entities and attributes essential to support the requirements of external files with optional version identification are illustrated in Figure 4 below:

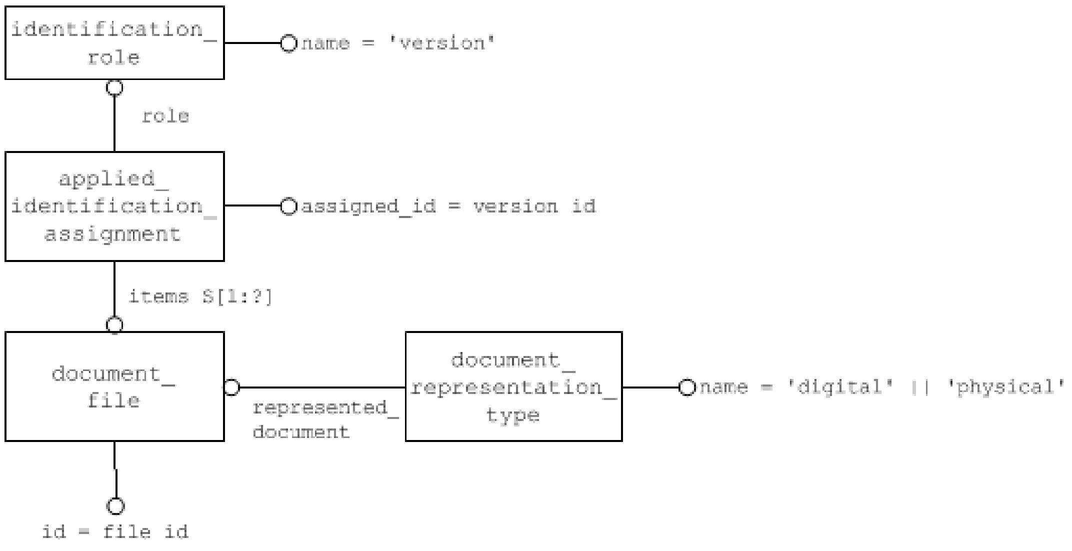

Figure 4: External File with Version Identification

5.2.2 Document File Version

5.2.2.1 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 `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'

ENTITY `applied_identi-fication_assignment`	**Attribute Population**	**Remarks**
`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.

5.2.2.2 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 `name` attribute should be assigned a value of 'version' in this usage.

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

Related Entities: There are no specific related entities.

Part 21 Example:

```
#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);
```

5.2.3 Document File Location

The three pieces of information that are relevant to unambiguously identify the target file are:

- A unique identifier (file id) for the file
- The file name, including the file extension
- The path to the target file

The path can be a URL or a directory path; in the latter case, it shall be given as a relative path from the source file to the target file, so that the reference still works should the entire package of files be moved to a different location (data exchange scenario).

Note: In the case where the external reference points to a physical document, the path can include information on the site, building, floor, room, and shelf the document is stored in.

5.2.3.1 applied_external_identification_assignment

This entity is a subtype of the entity `identification_assignment`. It allows the actual assignment of an `identification_assignment` entity to external items, in this case to a `document_file` entity representing an external file, including definition of the source.

Attributes

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

ENTITY `applied_external_identification_assignment`	**Attribute Population**	**Remarks**
`assigned_id`	type: identifier = string	contains the string identifying the file name of the external file.
`role`	type: entity = `identification_role`	for file location identification, this entity should have `name` attribute value = 'external document id and location'
`source`	type: entity = `external_source`	provides information about the path to the external file
`items`	type : entity = `document_file`	reference to the associated external file to which the name and location information is assigned.

Preprocessor Recommendations: In most cases, the unique identifier given in `document_file.file_id` (see section 5.2.1.1) will be a replicate of the file name, not at last to ensure compatibility with older implementations. There are however scenarios where this is not the case, where a database record id is given instead, or a company-wide unique identifier.

Postprocessor Recommendations:

- The name of the target file is to be expected in `applied_external_identification_assignment.assigned_id`.
- If this entity exists, but the `assigned_id` attribute is empty, evaluate the `source_id` attribute of the `external_source` referenced by the `source` attribute to check whether it also contains the file name.
- If `applied_external_identification_assignment` does not exist, evaluate `document_file.file_id`.

Related Entities: There are no specific related entities.

5.2.3.2 identification_role

This entity in this case indicates the role of id and location identification for an external file.

Attributes

- The `name` attribute is used to indicate that the related `identification_assignment` represents id and location identification for the associated external file.
- The `description` attribute is optional additional text.

ENTITY `identification_role`	**Attribute Population**	**Remarks**
`name`	type: identifier = string 'external document id and location'	Contains the string indicating that the related `identification_assignment` represents id and location information for an external file.
`description`	type: text = string	OPTIONAL

Preprocessor Recommendations: The `name` attribute should be assigned a value of 'external document id and location' in this usage.

Postprocessor Recommendations: The `name` attribute value 'external document id and location' should be interpreted as indication that the associated `applied_external_identification_assignment` represents the id and location information for the file.

Related Entities: There are no specific related entities.

5.2.3.3 external_source

This entity in this use case indicates the path information for an external file.

Attributes

- The `source_id` attribute indicates the path to the external file.

ENTITY `identification_role`	**Attribute Population**	**Remarks**
`source_id`	type: (identifier, message) = string	Contains the string indicating the external location (path) of the external file.

Preprocessor Recommendations: The path in `external_source.source_id` for digital files shall always be given relative to the source file. If an empty string is given, it means the target file will be stored in the same location (folder) as the source file

Postprocessor Recommendations: In cases where the `applied_exernal_identification_assignment.assigned_id` attribute is empty, evaluate the `source_id` attribute to check whether it also contains the file name.

Related Entities: There are no specific related entities.

5.2.4 Document File Association to Product Data

5.2.4.1 applied_document_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 (see section 5.5 below), external files need only the `applied_document_reference` entity to be related as a reference to other product data.

The EXPRESS entities and attributes used to support the requirements of external file association to product data are illustrated in Figure 5.

Part 21 Example:

```
#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);
```

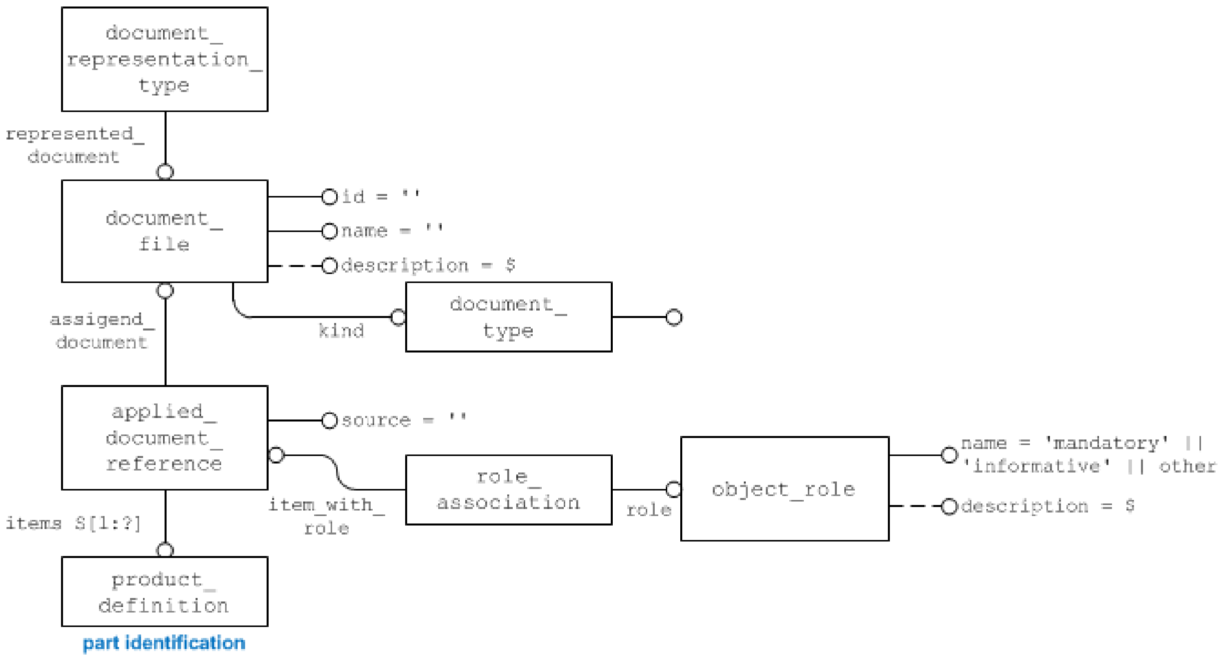

Figure 5: External File Association to Product Data

5.3 *"Document as Product" for Managed Documents*

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

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

These fundamental concepts are described for 'Part as Product' in the Recommended Practices for Basic PDM, Geometry and Structure. The following provides specifics on the distinction and additions for the 'Document as Product' approach.

'Document as Product' identifies a managed document object, e.g. 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.

As described in section 5.1 above, external files in STEP 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 5.4). 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`.

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:

- Document 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.

5.3.1 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 EXPRESS entities and attributes used to support identification of a document version are illustrated in Figure 6. Document version identification without an associated document view representation definition represents the minimum requirements for document master identification.

Part 21 Example:

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

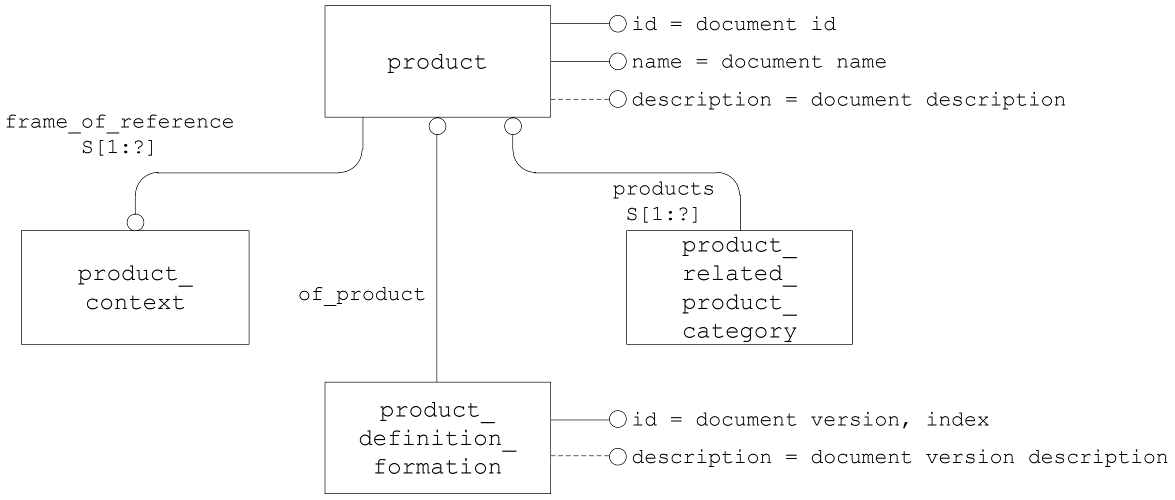

Figure 6: Minimum Document Identification

5.3.2 Context Information

Context information provides a scope and a necessary circumstance for document 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 respective application protocol and the application domain which provides a basis for the interpretation of all information represented in the product data exchange.

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 EXPRESS entities and attributes used to support the requirements of document identification with context information are illustrated in Figure 7.

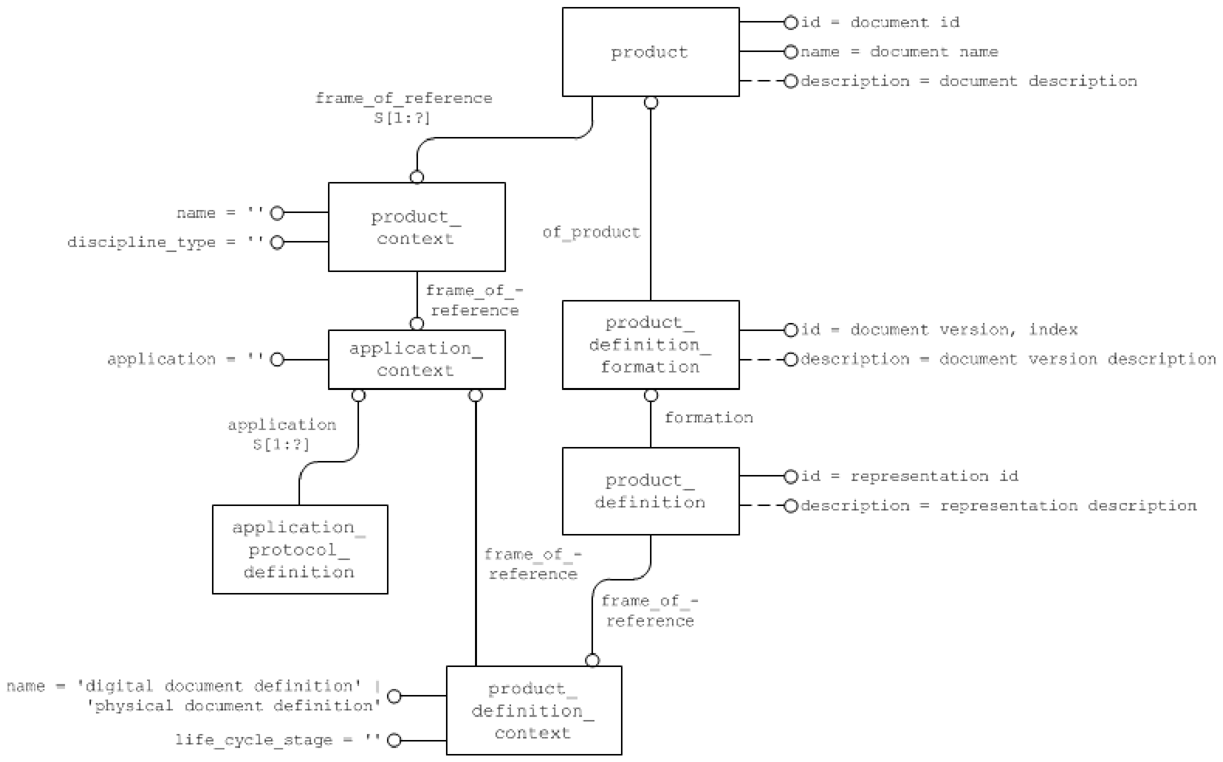
product
id = document id
name = document name
description = document description
frame_of_reference S[1:?]
product_context
name = ''
discipline_type = ''
of_product
frame_of_reference
application_context
application = ''
product_definition_formation
id = document version, index
description = document version description
formation
application S[1:?]
product_definition
id = representation id
description = representation description
application_protocol_definition
frame_of_reference
frame_of_reference
product_definition_context
name = 'digital document definition' | 'physical document definition'
life_cycle_stage = ''

Figure 7: Document Master with Context Information

5.3.2.1 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

ENTITY product_definition	**Attribute Population**	**Remarks**
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.

5.3.2.2 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.

Part 21 Example:

```
#240=APPLICATION_CONTEXT('automotive design');
#245=APPLICATION_PROTOCOL_DEFINITION('International Standard',
'automotive_design',2001,#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);
```

5.3.3 Type Classification

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

The EXPRESS entities and attributes used to support the complete requirements of document identification with context information and type classification are illustrated in Figure 8.

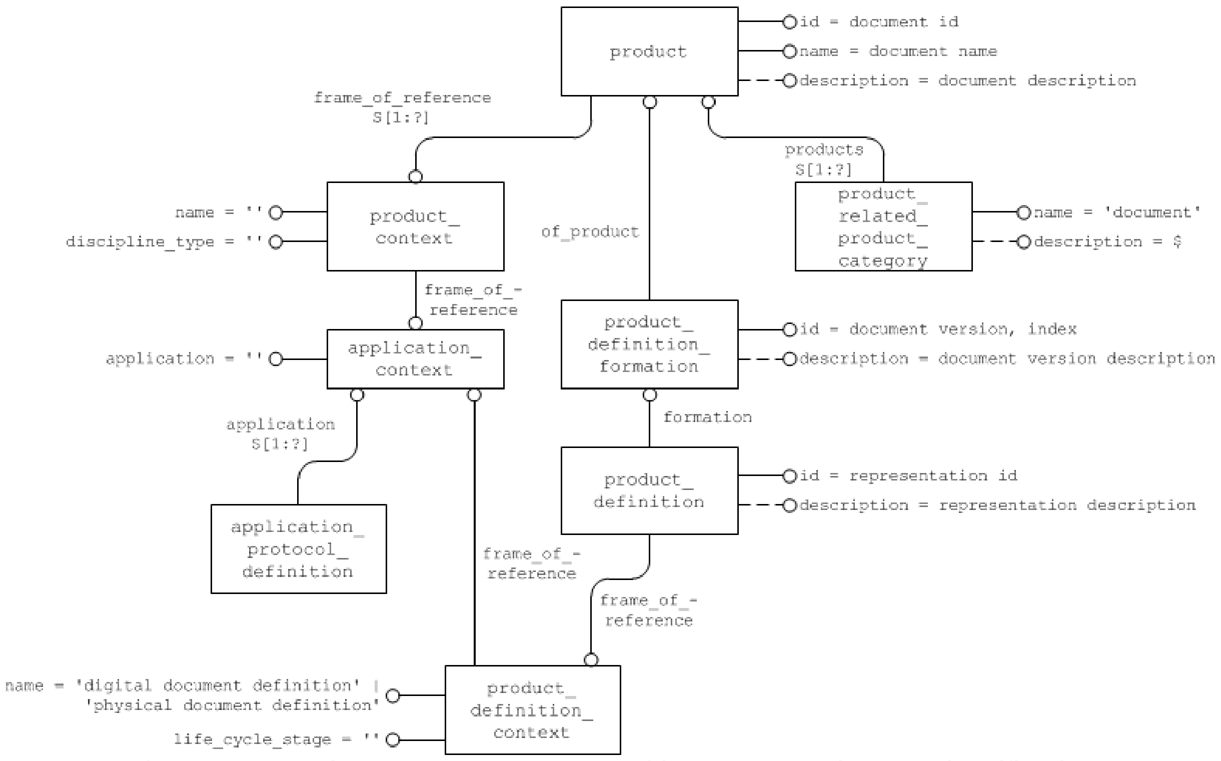
product
id = document id
name = document name
description = document description
frame_of_reference S[1:?]
products S[1:?]
product_context
name = ''
discipline_type = ''
of_product
product_related_product_category
name = 'document'
description = $
frame_of_-reference
application_context
application = ''
product_definition_formation
id = document version, index
description = document version description
formation
application S[1:?]
product_definition
id = representation id
description = representation description
application_protocol_definition
frame_of_-reference
frame_of_-reference
name = 'digital document definition' | 'physical document definition'
life_cycle_stage = ''
product_definition_context

Figure 8: Complete Document Master with Context and Type Classification

5.3.3.1 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.

Part 21 Example:

```
#240=APPLICATION_CONTEXT('');
#245=APPLICATION_PROTOCOL_DEFINITION('draft international
standard','ap242_managed_model_based_3d_engineering',2013,#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));
```

5.4 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 entity product_definition_with_associated_documents is used together with the identification of external files to support the requirement for the product as document with constituent external files.

The EXPRESS entities and attributes essential to support the requirements of document identification with associated constituent external files are illustrated in Figure 9.

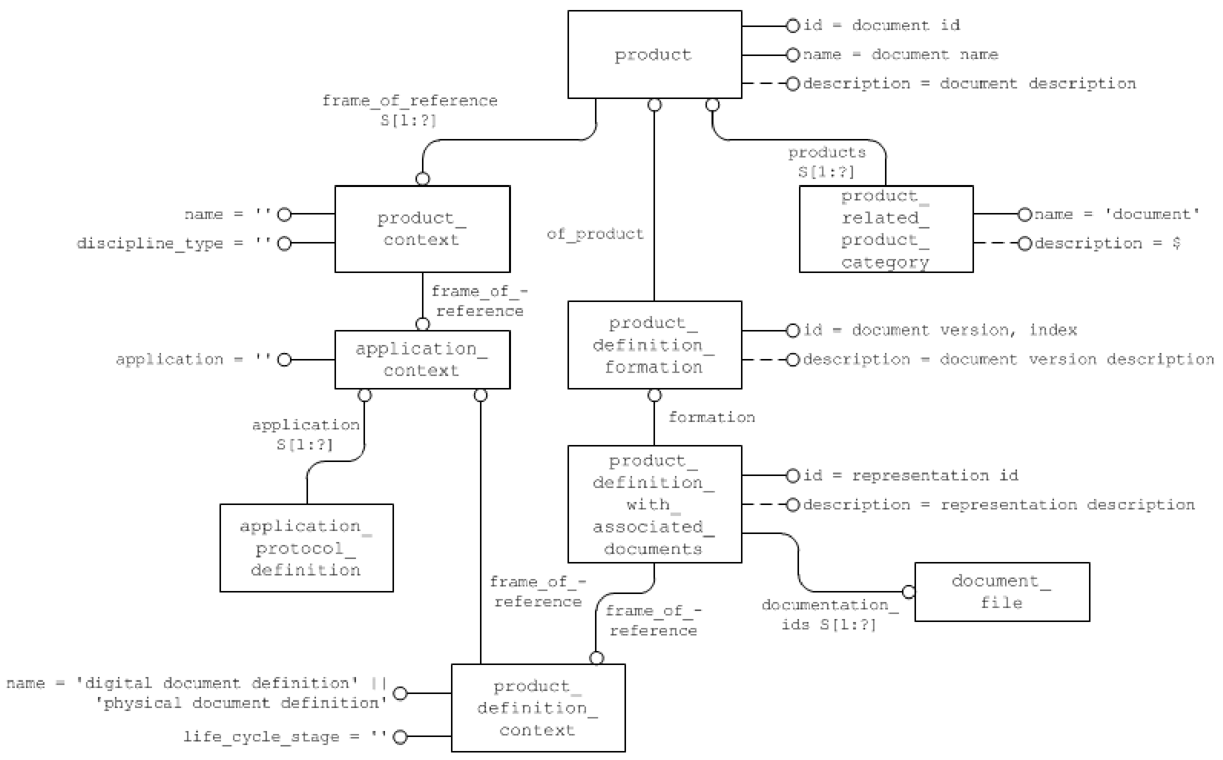
product
id = document id
name = document name
description = document description
frame_of_reference S[1:?]
products S[1:?]
product_ context
name = ''
discipline_type = ''
of_product
product_ related_ product_ category
name = 'document'
description = $
frame_of_- reference
application_ context
application = ''
product_ definition_ formation
id = document version, index
description = document version description
application S[1:?]
formation
product_ definition_ with_ associated_ documents
id = representation id
description = representation description
application_ protocol_ definition
frame_of_- reference
frame_of_- reference
documentation_ ids S[1:?]
document_ file
name = 'digital document definition' || 'physical document definition'
life_cycle_stage = ''
product_ definition_ context

Figure 9: Document Master with Constituent External Files

5.4.1 product_definition_with_associated_documents

The `product_definition_with_associated_documents` entity is a subtype of `pro-duct_definition`. It inherits the attributes `id`, `description`, `formation` and `frame_of_-reference` from the supertype. This entity is only used in the 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.

Part 21 Example:

```
#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','repre
sentation of version 3.2',#330,#250,(#340,#320));
```

5.5 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 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`.

5.5.1 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 EXPRESS entities and attributes essential to support the requirements of document association with other product data are shown in Figure 10.

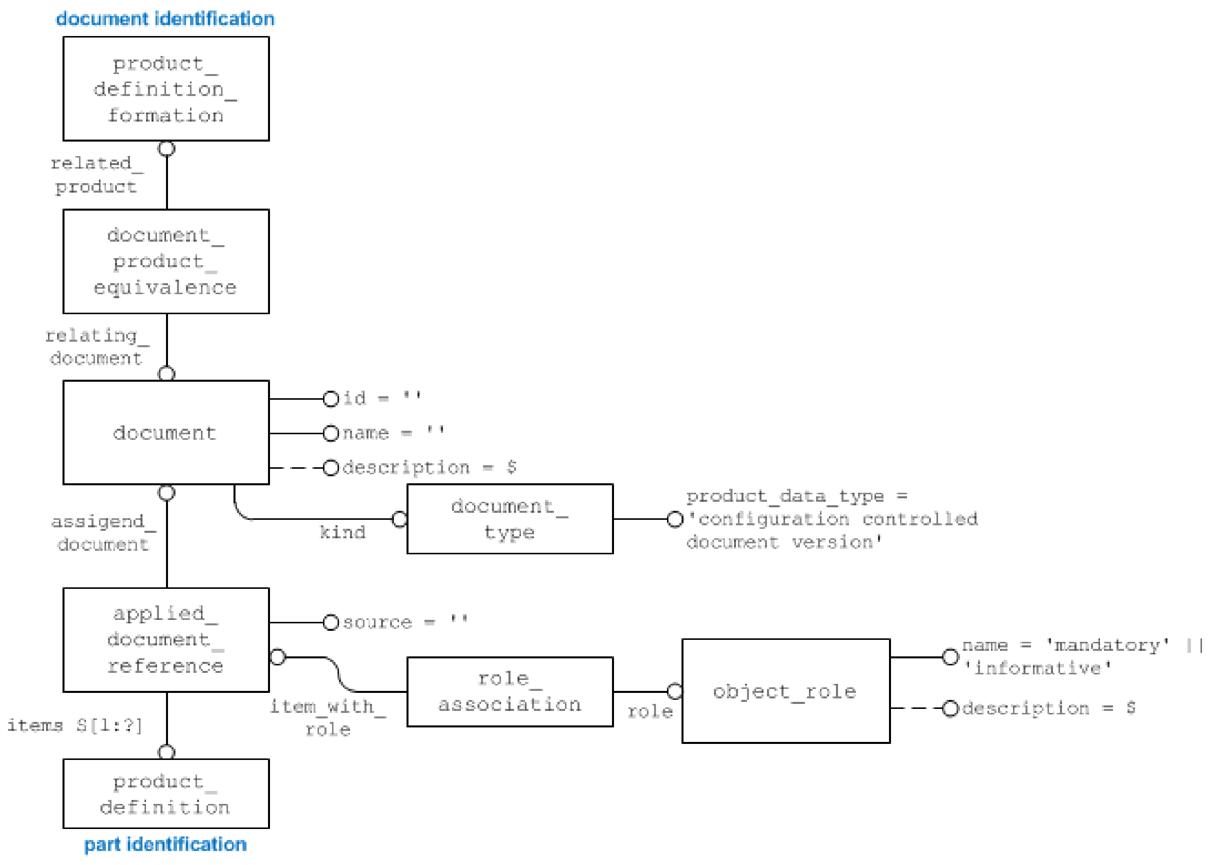

Figure 10: Document Association to Product Data

5.5.1.1 document_product_equivalence

This entity is a subtype of document_product_association relating a document to a 'part'. 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

ENTITY document_-product_equivalence	**Attribute Population**	**Remarks**
related_product	type: product_or_-definition_or_-formation (select)	specifies the component of the document master that is the point 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.

5.5.1.2 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.

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 section 5.1).

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.

5.5.1.3 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 usage 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.

5.5.1.4 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). Likewise, documents can be associated by versioned_action_request, executed_action or action_method.

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	

ENTITY `applied_-document_reference`	**Attribute Population**	**Remarks**
`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_refer-ence_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.

5.5.1.5 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 references the `role_select` type.
- 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.

5.5.1.6 object_role

This entity assigns a `role` to the associated document reference. For example, two generic roles that may be used are ‘mandatory’ and ‘informative’. Other role names are allowed when additional knowledge is required that provides specific semantics as to the role of the association between the document and the particular product data. Some of these additional role names are defined in the STEP APs. The two generic role names defined as examples are:

- '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', other	Other string values are valid based on requirements found in the APs.
`description`	type: text = string	OPTIONAL

Preprocessor Recommendations: The `name` attribute should have the value 'mandatory' or 'informative' if the required semantics to be captured is to indicate if the associated document reference is required or optional, respectively. APs may have other values for the `name` attribute that the preprocessor will need to handle.

Postprocessor Recommendations: The postprocessor should interpret `name` attribute as 'mandatory' or 'informative' when the required semantics being exchanged is to indicate if the associated document reference is required or optional, respectively. APs may have other values for the `name` attribute that the postprocessor will need to handle.

Related Entities: There are no specific related entities.

Part 21 Example:

```
#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',$);
```

5.5.2 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 EXPRESS entities and attributes used to support the requirements of constrained document or file association to product data are illustrated in Figure 11.

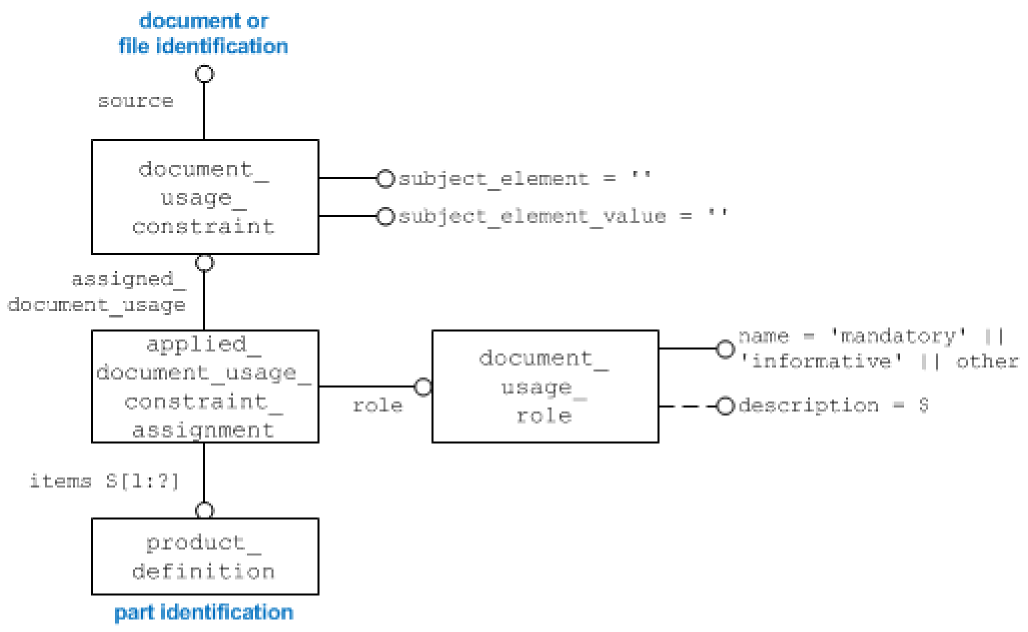

Figure 11: Constrained Document Association to Product Data

5.5.2.1 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.

5.5.2.2 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_usage_role`	
`items`	type : `documents_-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.

5.5.2.3 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', other	Other string values are valid based on requirements found in APs.
`description`	type : text = string	OPTIONAL

Preprocessor Recommendations: The `name` attribute should have the value 'mandatory' or 'informative' if the required semantics to be captured is to indicate if the associated document reference is required or optional, respectively. APs may have other values for the `name` attribute that the preprocessor will need to handle.

Postprocessor Recommendations: The postprocessor should interpret `name` attribute as 'mandatory' or 'informative' when the required semantics being exchanged is to indicate if the associated document reference is required or optional, respectively. APs may have other values for the `name` attribute that the postprocessor will need to handle.

Related Entities: There are no specific related entities.

Part 21 Example:

```
#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);
```

The following section combines the above-discussed concepts and segments in a complete example. The concepts instantiated in the example are illustrated in Figure 12.

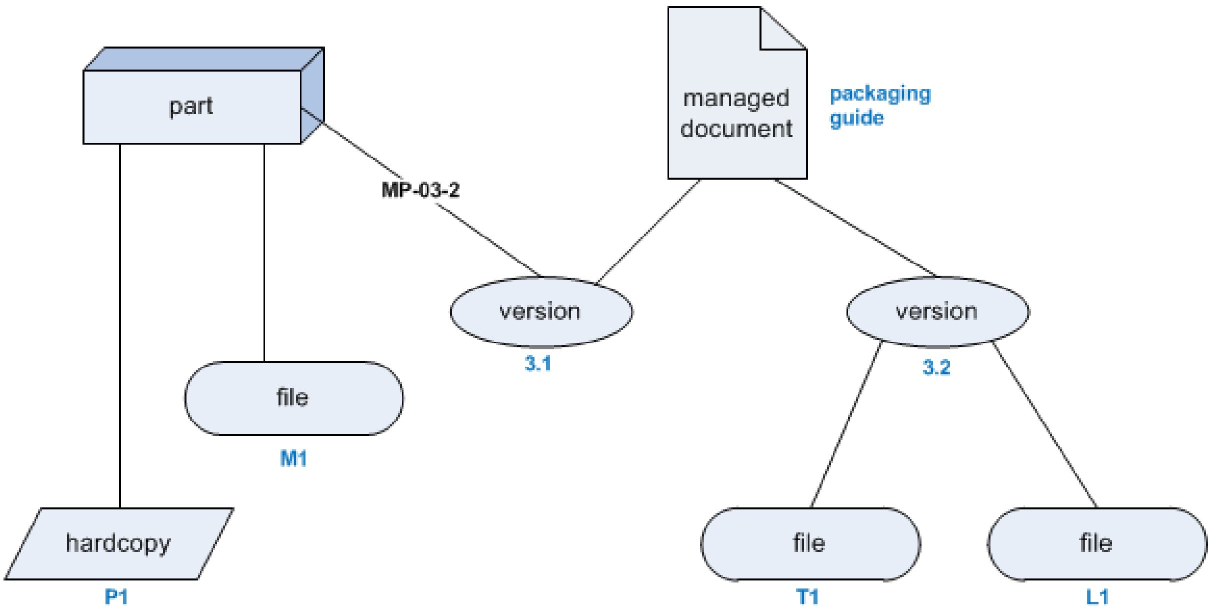

Figure 12: Schematic Overview of Complete Document and File Example

The EXPRESS entities and attributes used to support the requirements of constrained document or file association to product data are illustrated in Figure 13, Figure 14, Figure 15, and Figure 16.

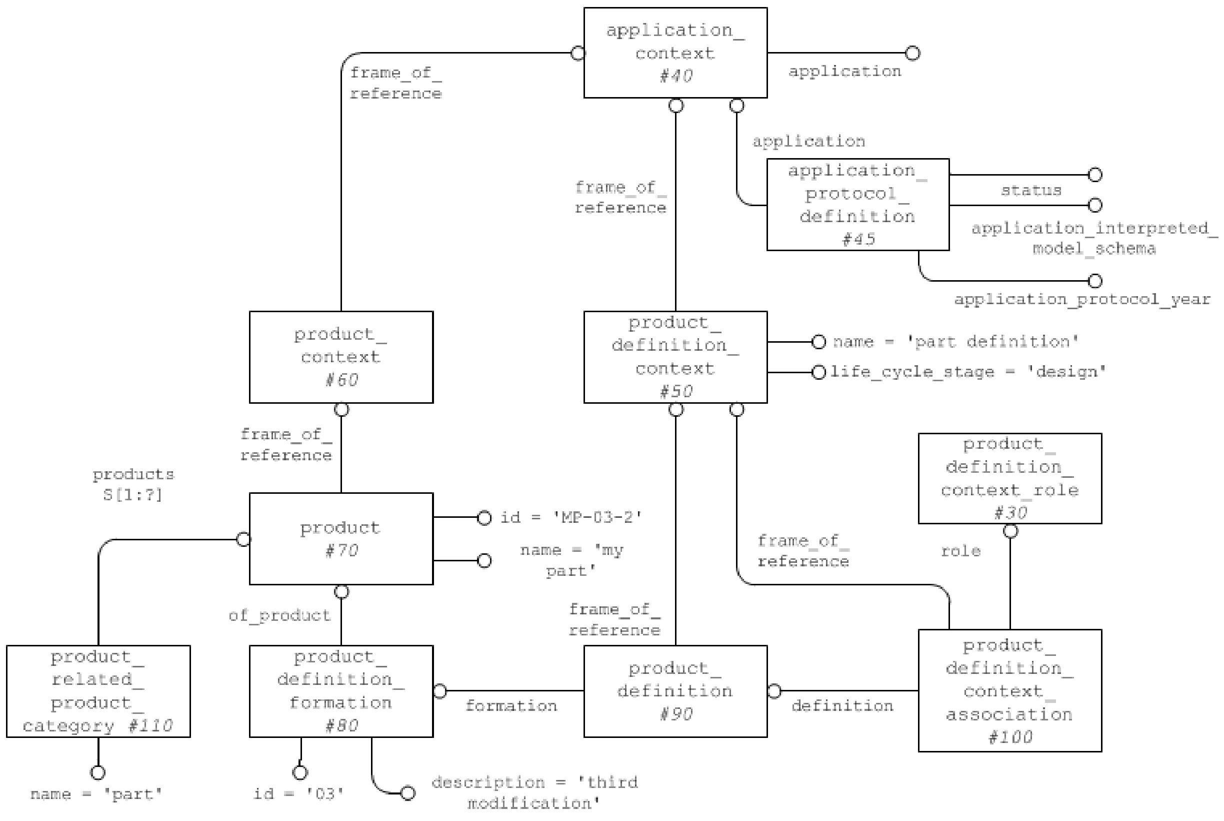
application_ context #40
frame_of_ reference
application
application
application_ protocol_ definition #45
status
application_interpreted_ model_schema
application_protocol_year
frame_of_ reference
product_ context #60
product_ definition_ context #50
name = 'part definition'
life_cycle_stage = 'design'
product_ definition_ context_role #30
frame_of_ reference
products S[1:?]
product #70
id = 'MP-03-2'
name = 'my part'
frame_of_ reference
role
of_product
frame_of_ reference
product_ related_ product_ category #110
product_ definition_ formation #80
formation
product_ definition #90
definition
product_ definition_ context_ association #100
name = 'part'
id = '03'
description = 'third modification'

Figure 13: Part Master

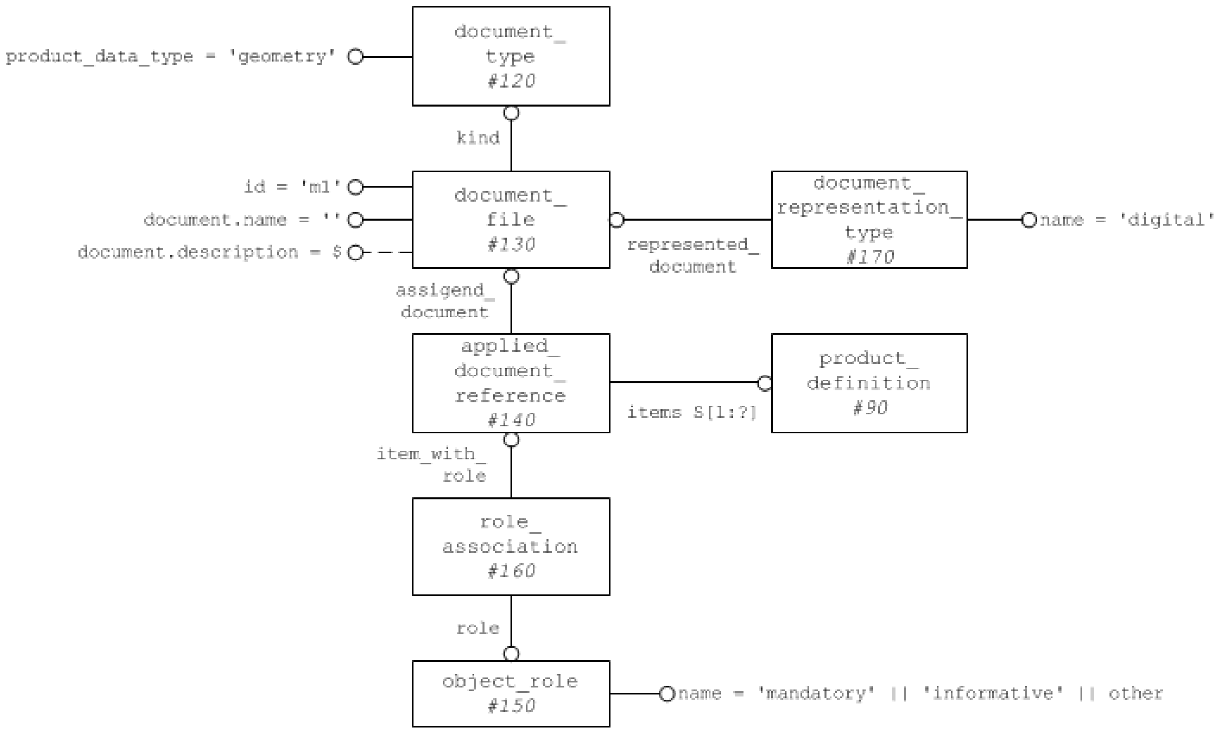

Figure 14: External File Reference to Part View Definition

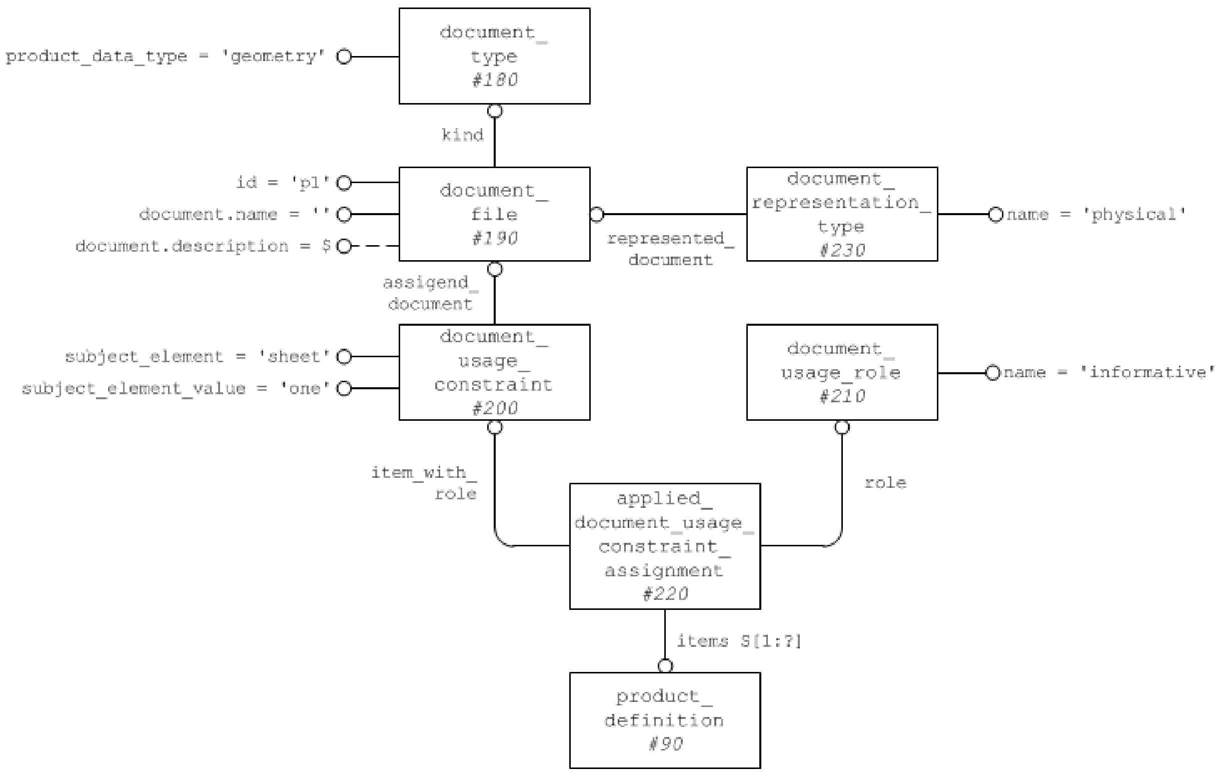
product_data_type = 'geometry'
document_ type #180
kind
id = 'p1'
document.name = ''
document.description = $
document_ file #190
represented_ document
document_ representation_ type #230
name = 'physical'
assigend_ document
subject_element = 'sheet'
subject_element_value = 'one'
document_ usage_ constraint #200
document_ usage_role #210
name = 'informative'
item_with_ role
role
applied_ document_usage_ constraint_ assignment #220
items S[1:?]
product_ definition #90

Figure 15: Constrained External File Reference to Part View Definition

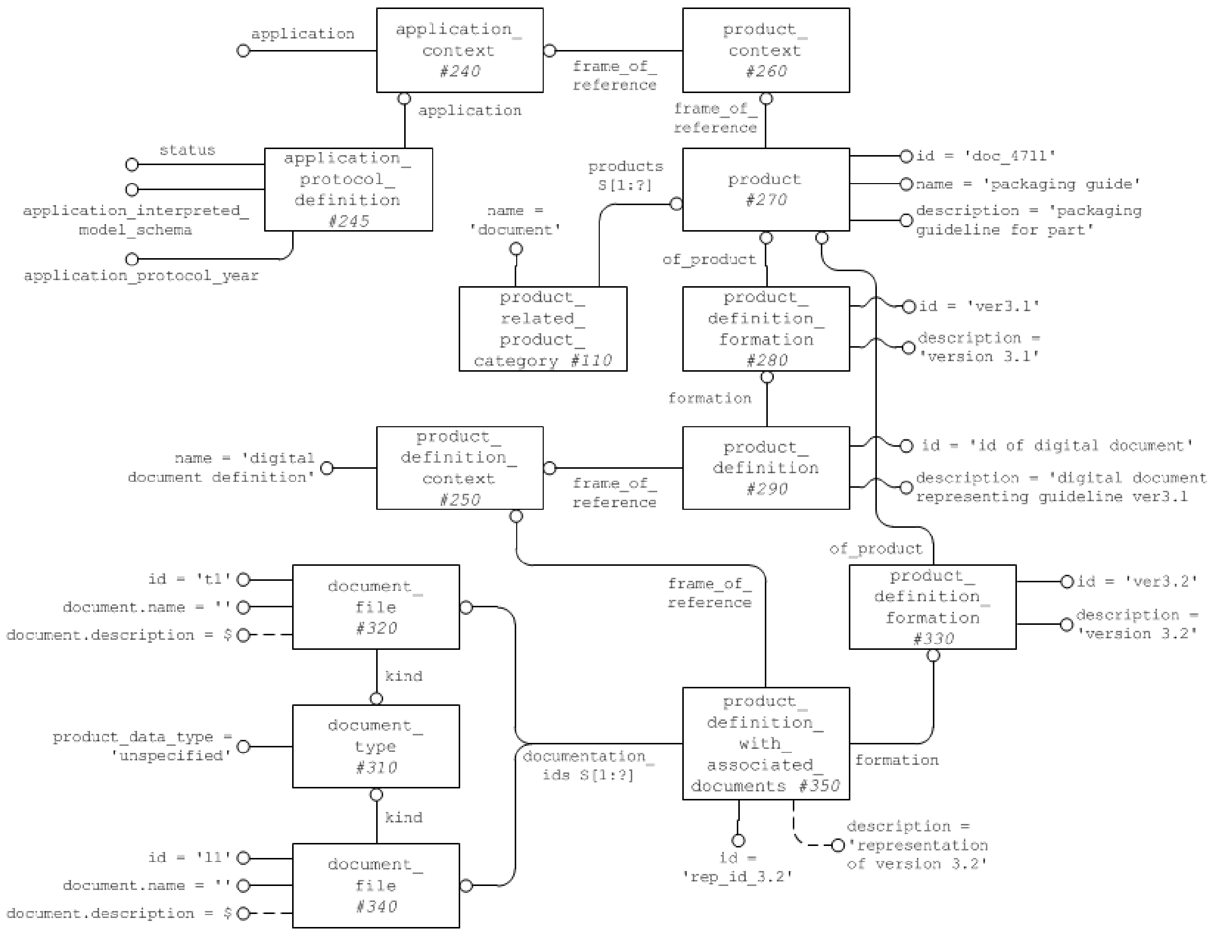
application
application_ context #240
frame_of_ reference
product_ context #260
application
frame_of_ reference
status
application_ protocol_ definition #245
application_interpreted_ model_schema
application_protocol_year
products S[1:?]
product #270
id = 'doc_4711'
name = 'packaging guide'
description = 'packaging guideline for part'
name = 'document'
of_product
product_ related_ product_ category #110
product_ definition_ formation #280
id = 'ver3.1'
description = 'version 3.1'
formation
name = 'digital document definition'
product_ definition_ context #250
frame_of_ reference
product_ definition #290
id = 'id of digital document'
description = 'digital document representing guideline ver3.1
of_product
id = 't1'
document.name = ''
document.description = $
document_ file #320
frame_of_ reference
product_ definition_ formation #330
id = 'ver3.2'
description = 'version 3.2'
kind
product_data_type = 'unspecified'
document_ type #310
documentation_ ids S[1:?]
product_ definition_ with_ associated_ documents #350
formation
kind
id = 'l1'
document.name = ''
document.description = $
document_ file #340
id = 'rep_id_3.2'
description = 'representation of version 3.2'

Figure 16: Document and Constituent File Association

Part 21 Example:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''), '2;1');
FILE_NAME('document_xample.stp', '2009-5-20 T12:39:1', (''), (), '',
   '', '');
FILE_SCHEMA(('AUTOMOTIVE_DESIGN { 1 0 10303 214 1 1 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('',$);
#40=APPLICATION_CONTEXT('automotive design');
#45=APPLICATION_PROTOCOL_DEFINITION('International Standard',
'automotive_design',2001,#40);
#50=PRODUCT_DEFINITION_CONTEXT('part definition',#40,'design');
#60=PRODUCT_CONTEXT('',#40,'');
#70=PRODUCT('MP-03-2','my part',$,(#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 assignment 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','representat
ion 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;
```

6 External References

The approach presented here is the so-called External References mechanism. It can in general be used to reference any kind of external information, ranging from a STEP file with part geometry to native CAD models to office and standards documents, which may be stored as digital files or even in a physical way, e.g. a printout.

In the scope of this document, however, External References will always be described as references from one file (a STEP AP203, AP214 or AP242 file) to another digital file. In the examples used, the target file is a part geometry file in STEP format, but the described concepts apply to other target file formats as well.

6.1 External File Reference

Externally defined geometry is identified as an external representation of the part shape that is related to the product. The external part shape is an external geometric model related to the product, and is referenced as an external geometry file. Specifics of external file identification are described in section 5.1. The external file that contains the detailed geometry is attached to the product 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`

An illustration of this structure is given in Figure 17.

It is recommended that `applied_document_reference` be used, in general, to relate an external file representing the geometry to the `product_definition` of a part. If the external file representing the geometry exists alone as an unmanaged external file reference, then the document reference should be applied directly to the `document_file`.

If the external file representing the geometry exists as a constituent file of a managed document (using the ‘Document as Product’ approach), the approach described in section 6.2 shall be used.

In the case the externally referenced file is another STEP file, it is clear that the entity chain from `product` down to `shape_representation` (on the left hand side of Figure 17) is a placeholder for the actual definition of the part and its shape given in the target file. This means that the relevant attributes have to match. These are: `product.id`, `product_definition_formation.id`, `product_definition.id`, and – if given, as required e.g. by AP214, the `organization.name` that is assigned to the `product` with the role ‘id owner’.

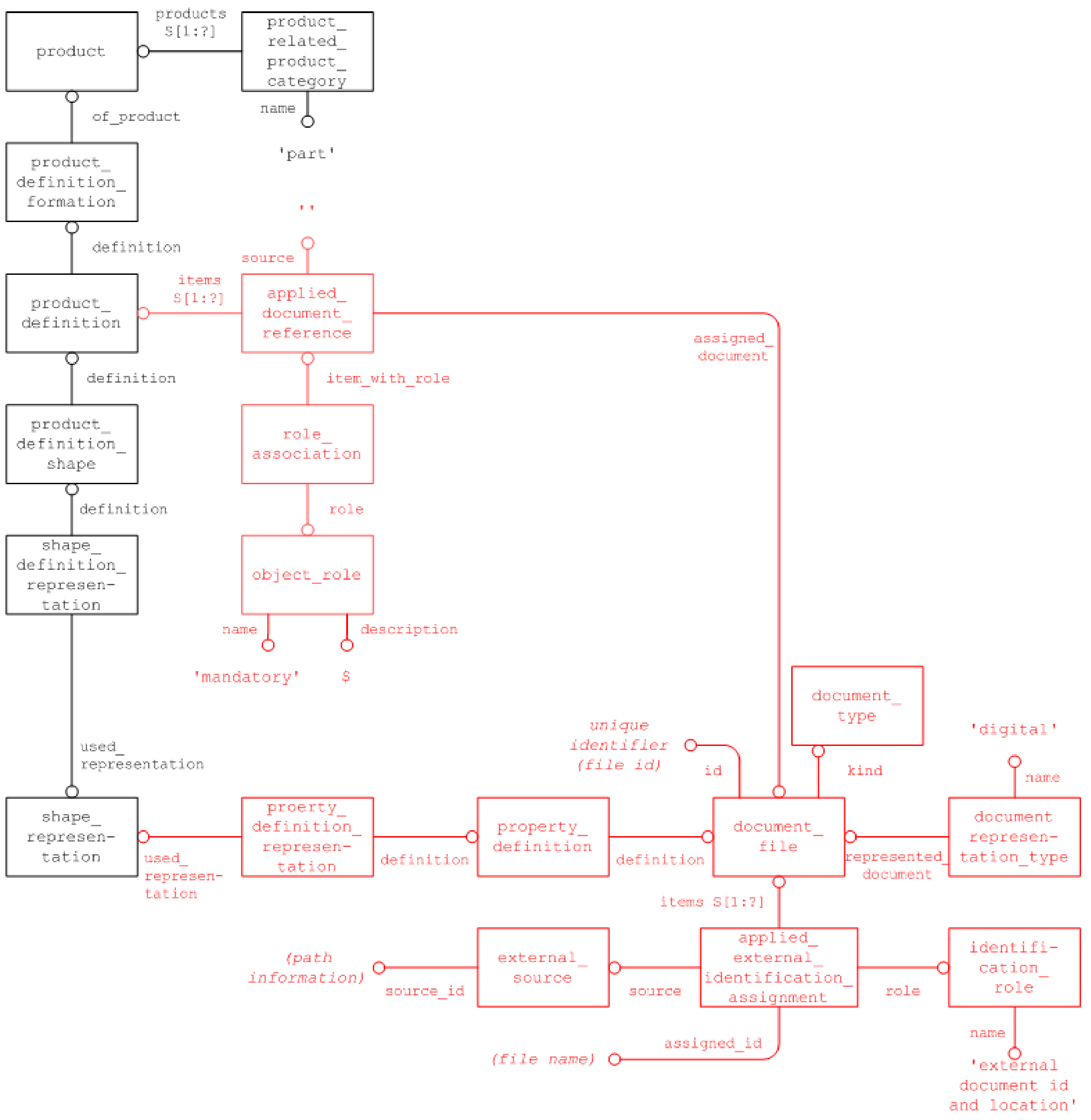

Figure 17: Instance diagram for External References

Part 21 Example:

```
/* Entities #200 to #230 define 'part1'                                  */
#200 = PRODUCT('p1', 'part1', '', (#20));
#210 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#200));
#220 = PRODUCT_DEFINITION_FORMATION('1', '', #200);
#230 = PRODUCT_DEFINITION('p1_d','part1 shape', #220, #215);
/* 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('external document id and location','access
context');
#582 = APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT('part1_geometry.stp',
#581,#583,(#576));
#583 = EXTERNAL_SOURCE('./part_dir/');
#584 = PROPERTY_DEFINITION('external definition',$,#576);
#585 = PROPERTY_DEFINITION_REPRESENTATION(#584,#520);
```

6.2 Document Management Information

In scenarios where files are exchanged between Product Data Management (PDM) systems, additional document management information may be valuable, or even required, to ensure the exchanged files can be handled correctly by the receiving system. This is the case when not only the products described in the files are being managed (i.e. versioned), but also the files containing the definitions are managed as well, e.g. when exchanged files are being stored persistently on the receiving side. It is important to point out that the file version and the version of the product described in that file do not necessarily have to be identical.

To convey this data, the 'Document as Product' approach is used in STEP. It is described in detail in sections 5.3, 5.4 and 5.5 above and extends the External Reference mechanism as presented before by introducing a number of additional entities, which are inserted into the existing structure. Compare Figure 18 below to Figure 17 above for an illustration of the extended structure.

For CAD systems importing a file with document management information, the additional data may not be relevant. However, to ensure interoperability, CAD systems need to be able to navigate the extended structure to access the required pieces of information. Since `applied_document_assignment` no longer points directly to the `document_file` if document management information is present, the path via `document`, `document_product_-equivalence`, `product_definition_formation` and `product_definition_with_-associated_documents` needs to be followed instead to get to the `document_file`. See Figure 18 for details.

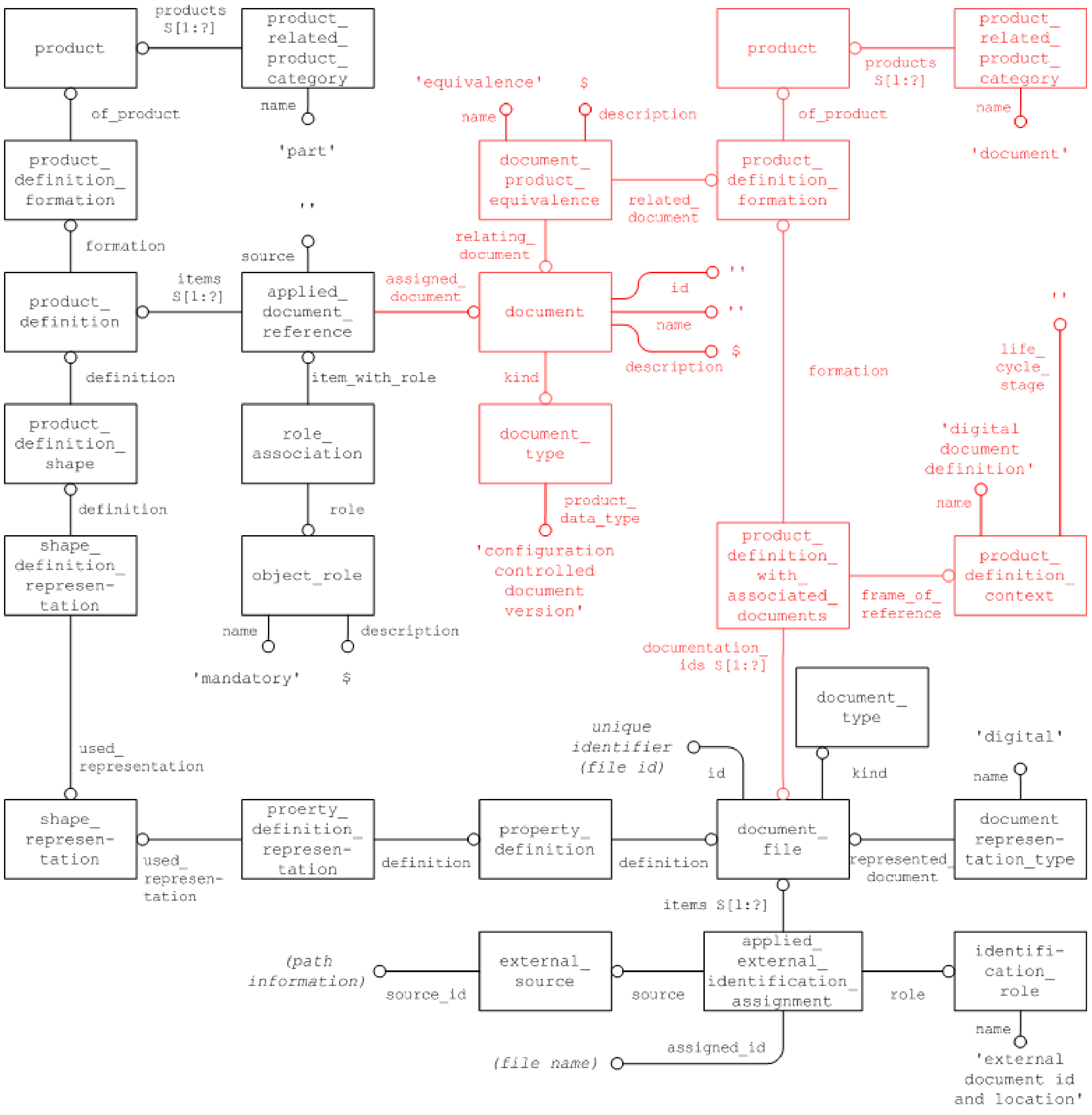

Figure 18: Instance diagram for External References with Document Management Information

<u>Part 21 Example:</u>

```
/* Entities #260 to #290 define 'part2'                            */
#260 = PRODUCT('p2', 'part2', '', (#20));
#270 = PRODUCT_RELATED_PRODUCT_CATEGORY('part', '', (#260));
#280 = PRODUCT_DEFINITION_FORMATION('1', '', #260);
#290 = PRODUCT_DEFINITION('p2_d','part2 shape', #280, #275);
/* 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, '', $);
#480 = DOCUMENT_REPRESENTATION_TYPE('digital',#475);
#485 = DOCUMENT_TYPE('');
#490 = IDENTIFICATION_ROLE('exernal document id and location','access
context');
#495 = APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT('part2_geometry.stp',
#490, #497,(#475));
#497 = EXTERNAL_SOURCE('./part_dir/');
/* Entities #600 to #660 define the geometric model for 'part2'          */
#600 = PRODUCT_DEFINITION_SHAPE('shape_p2',$,#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);
```

6.3 Document and File Properties

Document properties can be associated to representations of documents as well as to individual files. This means the properties described below can be used for simple External File References (section 6.1) as well as for references to Managed Documents (section 6.2).

If document properties are assigned to representations of (managed) 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.

6.3.1 Product Definition or Document Representation

The entity `product_definition` or `document_file` is used as target for document properties:

- Properties assigned to a (managed) document representation (i.e. the `product_-definition_with_associatied_documents` shown on the right hand side of Figure 18) apply to all document files associated with that document representation (see section 5.4).
- Properties assigned to a document file (i.e. the `document_file` at the bottom of both Figure 17 and Figure 18) apply to that file only.

The general schema for the assignment of properties to document representation (i.e., `pro-duct_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.

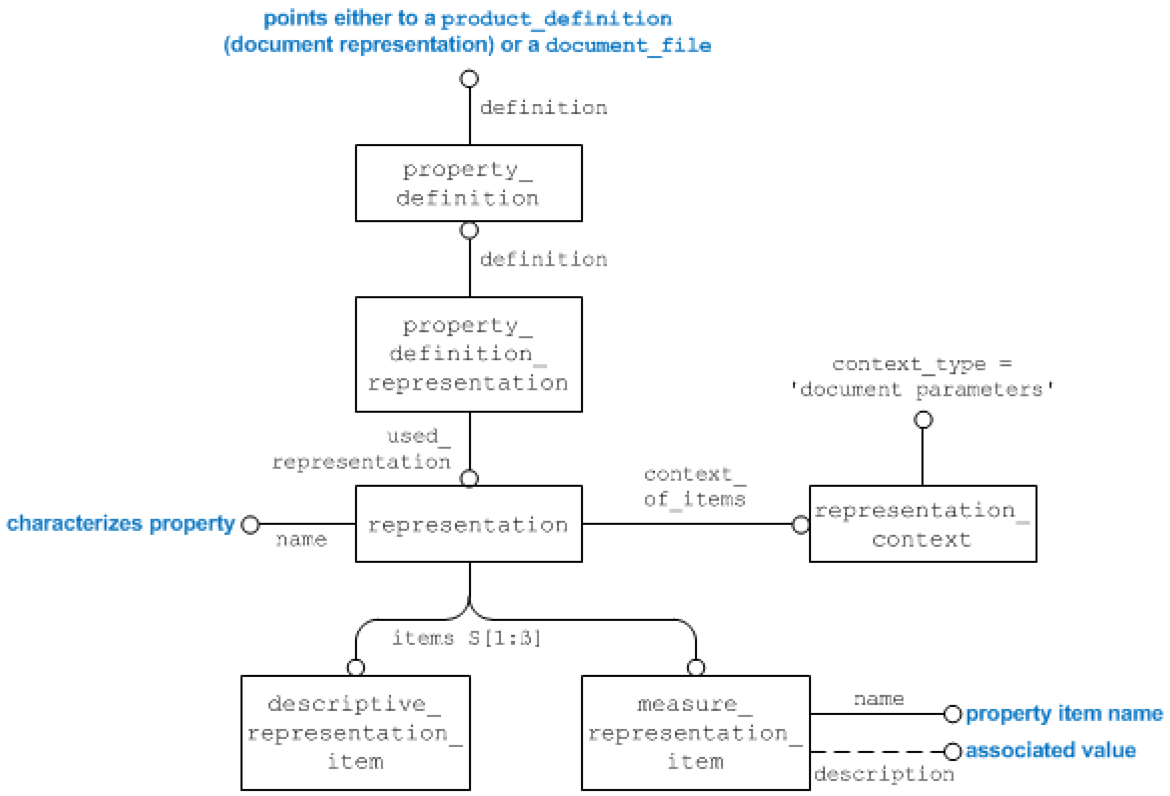
points either to a product_definition (document representation) or a document_file
definition
property_definition
definition
property_definition_representation
used_representation
context_type = 'document parameters'
characterizes property
name
representation
context_of_items
representation_context
items S[1:3]
descriptive_representation_item
measure_representation_item
name
property item name
associated value
description

Figure 19 : General pattern for the association of document properties

6.3.1.1 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

ENTITY `property_definition`	**Attribute Population**	**Remarks**
`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 specified.

Related Entities: None specified.

6.3.1.2 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 specified.

Postprocessor Recommendations: None specified.

Related Entities: None specified.

6.3.1.3 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 OR measure_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'

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 specified.

6.3.1.4 representation_context

A representation_context 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 specified.

Related Entities: None specified.

6.3.1.5 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 prop-erty
description	Type: text = STRING	the description is the value associated with the represent-ation item in textual form

Preprocessor Recommendations: None specified.

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.

6.3.2 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'

Figure 20 below illustrates the entity structure for this property:

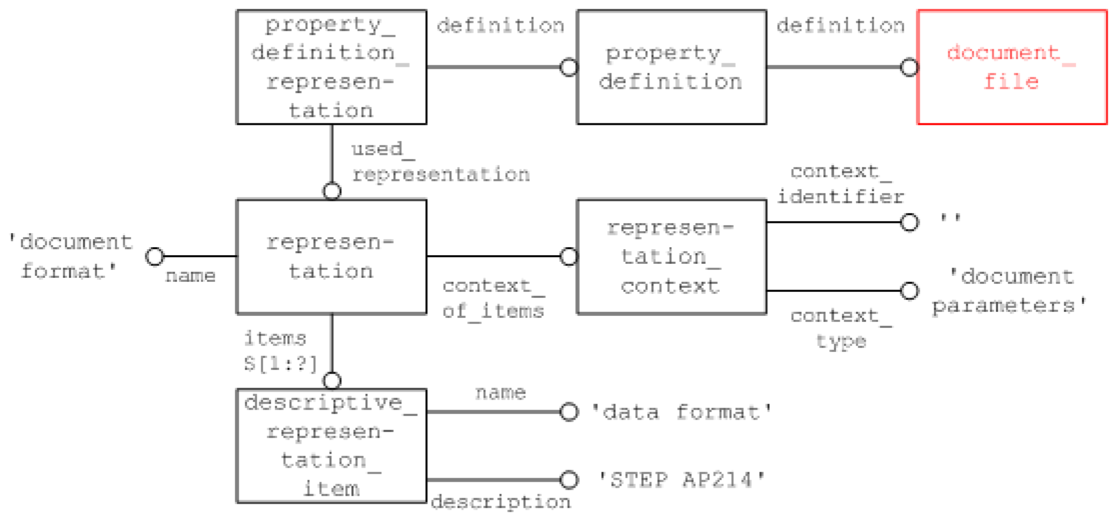

Figure 20: Instance diagram for Document Format Properties

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_represen-tation_item.description`**
the convention that was used to structure the information in the characterized object	'data format'	Where applicable the values listed in Table 2 below shall be used
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 file format is stored in the `description` of the `descriptive_representation_-item` with `name` 'data format'. The following table summarizes the recommended values:

Value	**Target file format is…**
'STEP AP203'	First edition of AP203 (1994). No longer to be used on export; but may occur on import of old files.
'STEP AP203E2'	Second edition of AP203 (2009).
'STEP AP214'	AP214 first edition IS (2001) full schema.
'STEP AP214 CC06'	AP214 conformance class 06 (product structure only). Such a file will contain no geometry.
'STEP AP214E3'	AP214 third edition (2010) full schema.
'STEP AP242'	AP242 first edition (2013) full schema.

Table 2: Recommended Document Format Property Values

Note that the document format property only provides information about the file format; not about the file contents. For instance, a referenced AP214 file may contain single-part geometry without further structure, the definition of a subordinate assembly structure hierarchy level without any geometry, or the complete definition of a component including sub-assembly structure and geometry. See section 6.3.3 for how to describe the external file contents.

Note also that the allowed values for the document format property are not limited to the list given in Table 2. If the referenced file is e.g. a native CAD file, the document format property may provide information about system name, major and minor release, patch level, etc.

Part 21 Example:

```
/* 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), #200);
#170 = DESCRIPTIVE_REPRESENTATION_ITEM('data format', 'STEP AP214');
#200 = REPRESENTATION_CONTEXT('', 'document parameters');
```

6.3.3 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;

Requirement	descriptive_-representation_-item.name	descriptive_representation_item. description
		• 'solid and surface model': The document contains a 3D shape model in surface and advanced boundary representation; • ‘tessellated geometry’: The document contains a simplified geometric representation; • 'assembly': The document contains an assembly structure with reference to the assembled components and their transformation matrices; • 'assembly with mating elements': The document contains an assembly structure including the mating components only, such as screws or rivets, with exact positioning information. This assembly representation is intended to be overlaid with the assembly structure for the main components; • '2D drawing': The document contains a technical drawing without 3D shape representation; • 'drawing derived from 3D data': The document contains a technical drawing that has been derived from a 3D shape model; • 'drawing related to 3D data': The document contains a technical drawing that visualizes a 3D shape model and possibly establishes associative links to the 3D shape model.
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'

Part 21 Example:

```
/* 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');
```

6.3.4 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 document creation 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'

Part 21 Example:

```
/* 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');
```

6.3.5 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 the 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'

6.3.5.1 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 instantiate `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 specified.

Related Entities: None specified.

Part 21 Example:

```
#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);
```

6.3.6 Document Source Property

This capability allows the specification of the 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, 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_assign-ment.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 instantiation diagram is shown below in Figure 21.

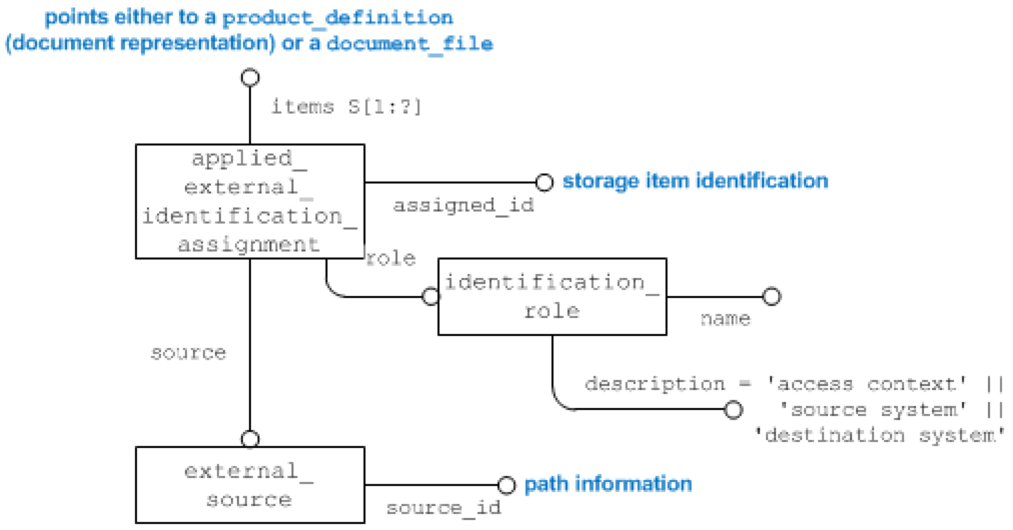
points either to a product_definition (document representation) or a document_file
items S[1:?]
applied_external_identification_assignment
assigned_id
storage item identification
role
identification_role
name
source
description = 'access context' || 'source system' || 'destination system'
external_source
source_id
path information

Figure 21 : Document Source Property

6.3.6.1 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 specified.

Related Entities: None specified.

6.3.6.2 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 = identificiation_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 specified.

Related Entities: None specified.

6.3.6.3 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 specified.

Postprocessor Recommendations: None specified.

Related Entities: None specified.

NOTE - Additional document properties can be defined on an individual basis.

6.3.7 Document Type Classification

STEP supports the assignment of document type information to documents and document files. Type classification of documents is discussed in Section 5.3.3. The other approach is outlined in the following subsections. In the case of an assignment to a document file, the type characteristics apply on an individual basis. STEP does not support the assignment of a classification system relative to which the classification has been done.

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.

6.3.7.1 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 specified.

Related Entities: This entity is required for each instance of the entity document.

Part 21 Example:

```
#80 = DOCUMENT_FILE('file1', '', $, #90, '', $);
#90 = DOCUMENT_TYPE('geometry');
#100 = DOCUMENT_REPRESENTATION_TYPE('digital', #80);
```

6.4 *Nested External References*

A special case in the context of External References is the exchange of assemblies, since the mechanism can be applied in different ways, as already indicated in the introduction. In the basic scenario, the entire assembly structure is defined in one structure file, and for each component part, there is an External Reference to the file containing the geometry definition. The following figure illustrates this based on the structure of the well-known “AS1” example:

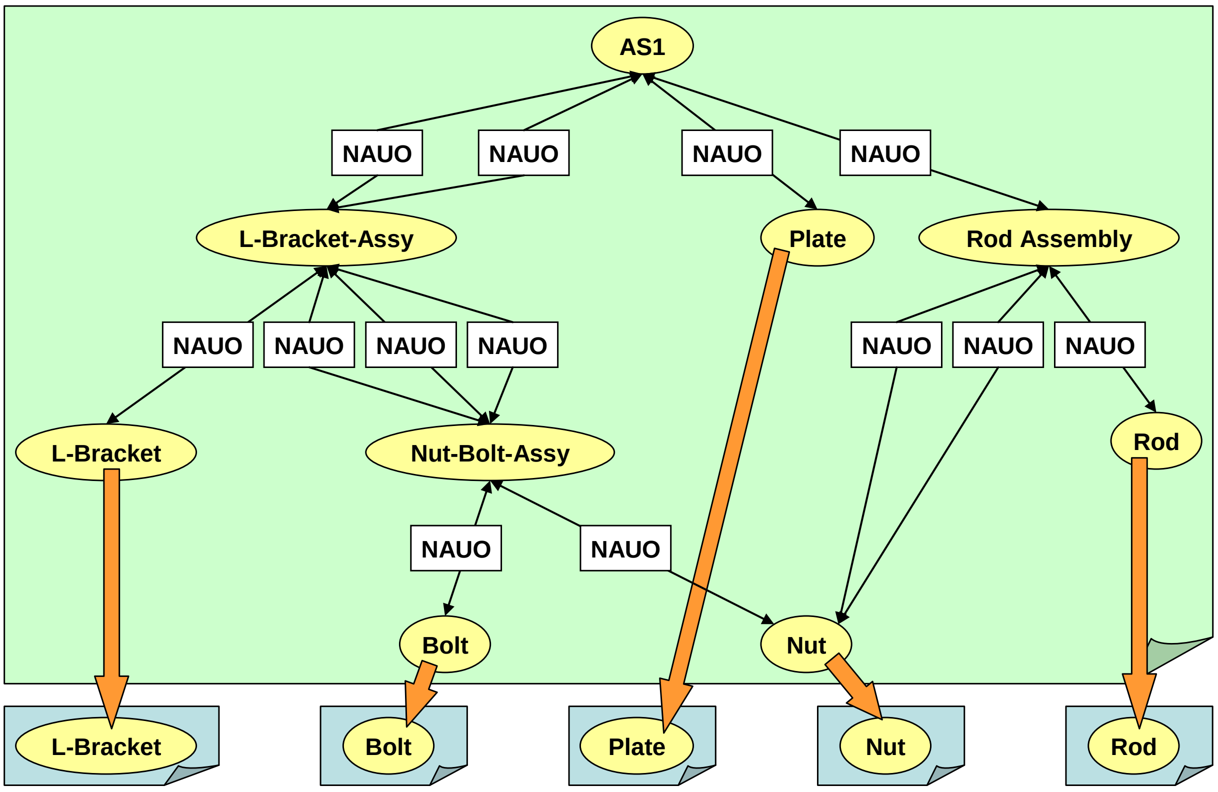

Figure 22: File Structure for Basic External References

It is important to note that even if a component part is used multiple times in the assembly, the corresponding file with the geometry definition is referenced only once. In Figure 22 above, the structure file is shown in green, the leaf-node geometry files in blue, and each of the thick orange arrows represent one External Reference.

There are many use cases which motivate the splitting of the assembly structure into several files; for instance, in contexts with large assemblies (e.g. aircraft, automotive, and ship manufacturing), there may be the need to exchange only a portion of the entire structure. Though there is no strict rule requiring this, the usual approach is to have one assembly structure hierarchy level per file, i.e. each structure file contains a root or intermediate node and its immediate children, which then reference the subordinate structure or part files.

Figure 23 below illustrates this by splitting the assembly structure shown in Figure 22 into four structure files. Other than that, the same rules apply as in the basic case. Note that the file containing the geometry of the “Nut” part is now referenced twice, since this part occurs in two different sub-assemblies.

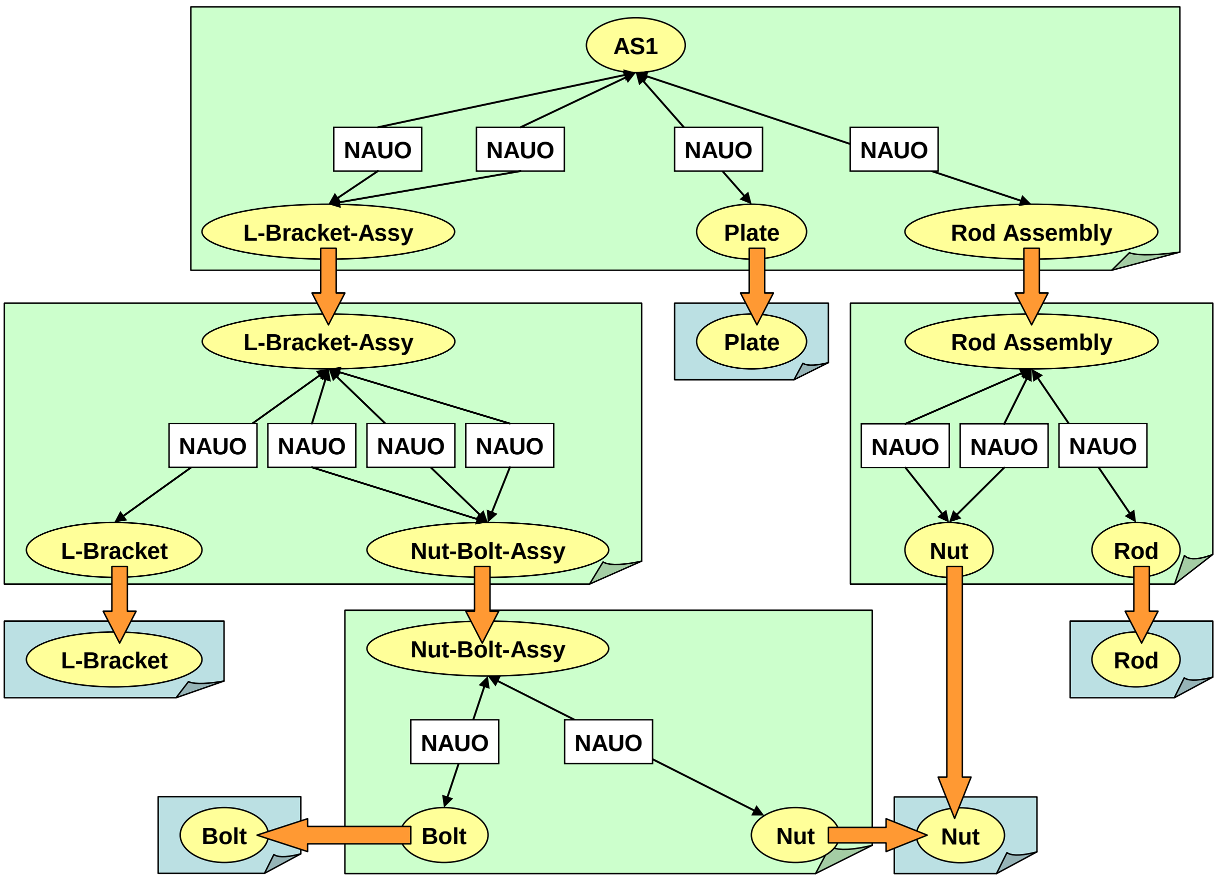

Figure 23: File Structure for Nested External References

7 External Element Reference (EER)

While the External Reference mechanism as described in section 5 above is well-established in industry use and supported by a range of STEP application protocols, it always treats the referenced files as a “black box”. Recently, a number of use cases emerged that have a requirement to associate information between elements of different components – or instances thereof – in an assembly. This is currently possible only if the entire model definition is contained in one file.

The requirement hence was to create a mechanism that allows for unambiguously pointing to a specific element inside an external file, so that data in the assembly context can be used in a scenario where external references are used as well. In the context of the development of

AP242, a first stage of the EER mechanism has been realized based on use cases for Assembly Product Manufacturing Information (PMI), Composites, and Validation Properties.

The following sections will describe and define the generic concepts for EER. How External Element References are used in a particular context will be described in the respective domain or use-case specific recommended practices documents.

7.1 Basic Concepts

The idea behind EER is basically the same as with section references in HTML. In the target file, an anchor with a unique name is defined at the desired location in that page. The source file then calls upon it by referencing “target.html#anchor”.

With AP242, this mechanism was carried over to STEP, and is supported in both the classic STEP file (Part 21) and the XML (Part 28) representation. It is generic in the way that it does not limit the target file to be in AP242 format either. The target file can be of any format as long as it supports the definition of such anchors, and the exporting system has sufficient access to the data to determine the correct anchor. Even “navigation instructions” can be given to navigate along a series of such anchors.

However, these Recommended Practices will focus on the scenario where all involved files are AP242 Part 21 files. Other scenarios may be described in specific Recommended Practices.

The stability of the EER links across files requires special attention when the model is updated, and some or all of the resulting STEP files are being updated. In use cases where source-target file pairs are being created simultaneously, this does not matter. But in scenarios involving assemblies, it needs to be ensured that, if a target file changes, the source files referencing are updated as needed.

Such updates might become necessary when:

- A new version of a component part is exchanged and the file name changes (this is independent of EER and has to be treated as was done with classic External References)
- New data is added to the assembly which requires new links.

In the second case, it is clear that the files which contain that new data need updating as well. To make sure that no other links are broken, it needs to be ensured that:

- The name (unique ID) of existing anchors does not change
- Existing anchors are not removed from the target file, unless it can be ensured that there is no other file still using it.

Note that the addition of further links to a target file will have no effect on the already existing ones.

One topic for which no recommendation can be given yet is which incoming references to create in a STEP file. It is certainly not feasible (or meaningful) to create an anchor for every entity in a file. The other extreme is to create only those anchors which are needed in the context of the entire current model. Experience is needed to answer this. It may also turn out that there is a meaningful default set of anchors to be created. This section of this document will be updated in the future, when sufficient experience with EER has been gathered.

7.2 Global Unique Identifiers (GUIDs)

This section intends to provide a starting point for how anchor names for external element references can be created in a unique way, based on established definitions used in the building industry, which uses a data model derived from STEP many years ago.

Here is some information about GUIDs for IFC and CIS/2. A GUID is required in IFC for all child entities of IfcRoot:

- http://www.buildingsmart-tech.org/ifc/IFC2x4/rc3/html/link/ifcroot.htm

This is the IFC documentation for a GUID:

- http://www.buildingsmart-tech.org/implementation/get-started/ifc-guid/

In IFC, the 36 character GUID is compressed to 22 characters to reduce file size. This was an issue back in the late 90's when IFC was being developed.

In CIS/2, the use of a GUID is optional. A GUID can be assigned to a part, cartesian point, direction, length, material, anything, etc. with the `managed_data_item` entity.

- http://cic.nist.gov/vrml/cis/lpm6/index.html

One problem seen with GUIDs in CIS/2 and IFC files is that some software vendors cheat when generating a GUID. A GUID should appear as a completely random string of characters. One GUID should not look like the next GUID. However, sometimes one GUID is generated and the next GUID only increments one of the characters by one, which is incorrect.

7.3 External Element References to Representations and Representation Items

The most generic part of the EER mechanism allows for referencing `representations` and `representation_items` in an external file. A typical scenario for this is where there is one file containing all the basic information about a product (e.g. the geometry), and a second file that provides additional information for semantics or validation, which points back into the base file.

7.3.1 Target File

For each incoming reference, an anchor needs to be defined in the target file. This is done in the header section of the file, using user-defined entities as defined in Part 21. Here's the corresponding definition taken from ISO 10303-21:2002, section 8.3:

8.3 User-defined header section entities

User-defined header section entity instances may be placed in the header section with specific restrictions as listed below:

a) User-defined header section entity instances shall conform to the same syntax of all header section entity instances with the additional requirement that the first character of the keyword shall be an exclamation mark "!".

b) The attributes of user-defined header section entities shall have EXPRESS data types and shall be mapped to the header section as specified in clause 10.

EXAMPLE

```
HEADER;
    .
    .
FILE_SCHEMA(('GEOMETRY'));
!A_SPECIAL_ENTITY ('ABC',123); -----> USER DEFINED ENTITY
    .
    .
ENDSEC;
```

The agreed name for the entity to use in the given context is `EXTERNAL_ANCHOR`. The task of this entity is to "translate" the unique ID for the external element reference to the actual instance number of the target entity in the data section. So, if during an update of the file the instance number changes and the anchor is updated accordingly, the reference will still work since nothing has changed for the outside world.

Following is an example of how a Part 21 file with inbound references following this pattern would look:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION( ... );
FILE_NAME( ... );
FILE_SCHEMA(('AP242_MANAGED_MODEL_BASED_3D_ENGINEERING_MIM_LF'));
!EXTERNAL_ANCHOR('MB3OOTCM', #100);
!EXTERNAL_ANCHOR('DGKEM5F4', #125);
ENDSEC;
DATA;
… OTHER INSTANCE DATA …
#100=ADVANCED_BREP_SHAPE_REPRESENTATION( … normal data … );
#125=ADVANCED_FACE( … normal data … );
… OTHER INSTANCE DATA …
ENDSEC;
END-ISO-10303-21;
```

It is important to point out that for each entity instance in the data section, there shall be not more than one EXTERNAL_ANCHOR referencing it. And vice versa, each EXTERNAL_ANCHOR references exactly one entity instance in the data section.

When updating a STEP file, the ID string has to be maintained ('MB3OOTCM' in the example above), while the reference to the entity (#100) will be updated to point to the new number of this entity in the updated file. This means that the authoring systems have to be able to store information about these external anchors in their internal data model, so that none are missed and all are exported consistently when re-exporting a model that has external element references in it.

A big advantage of this approach is that incoming references can be added very easily to an already existing STEP file, e.g. by a third-party tool, since all that is needed is adding a line to the header section – no changes to the data section are required at all!

7.3.2 Source File

The most important task when creating external element references is to establish the correct context for the targeted item. That means for a representation_item, it needs to be declared within which representation it is being used, and for a representation, it needs to be known in which context (part, version, view…) it is being used. The context for the representation will be created in the source file.

In order to establish the context for an externally referenced representation_item, the representation it is contained in needs to be referenced as well. So for both entities, placeholders are needed in the source file, from which the external element references to the target file will be created, and which are used to include the externally referenced information in the definition of the source file contents.

The entities needed for this are:

- externally_defined_representation
- externally_defined_representation_item

They are joint subtypes of externally_defined_item and representation(_item), where the exernally_defined_representation is restricted to allow only externally_defined_representation_items in its set of items. If needed, they can be combined with specific representation(_item) subtypes as complex entities if needed.

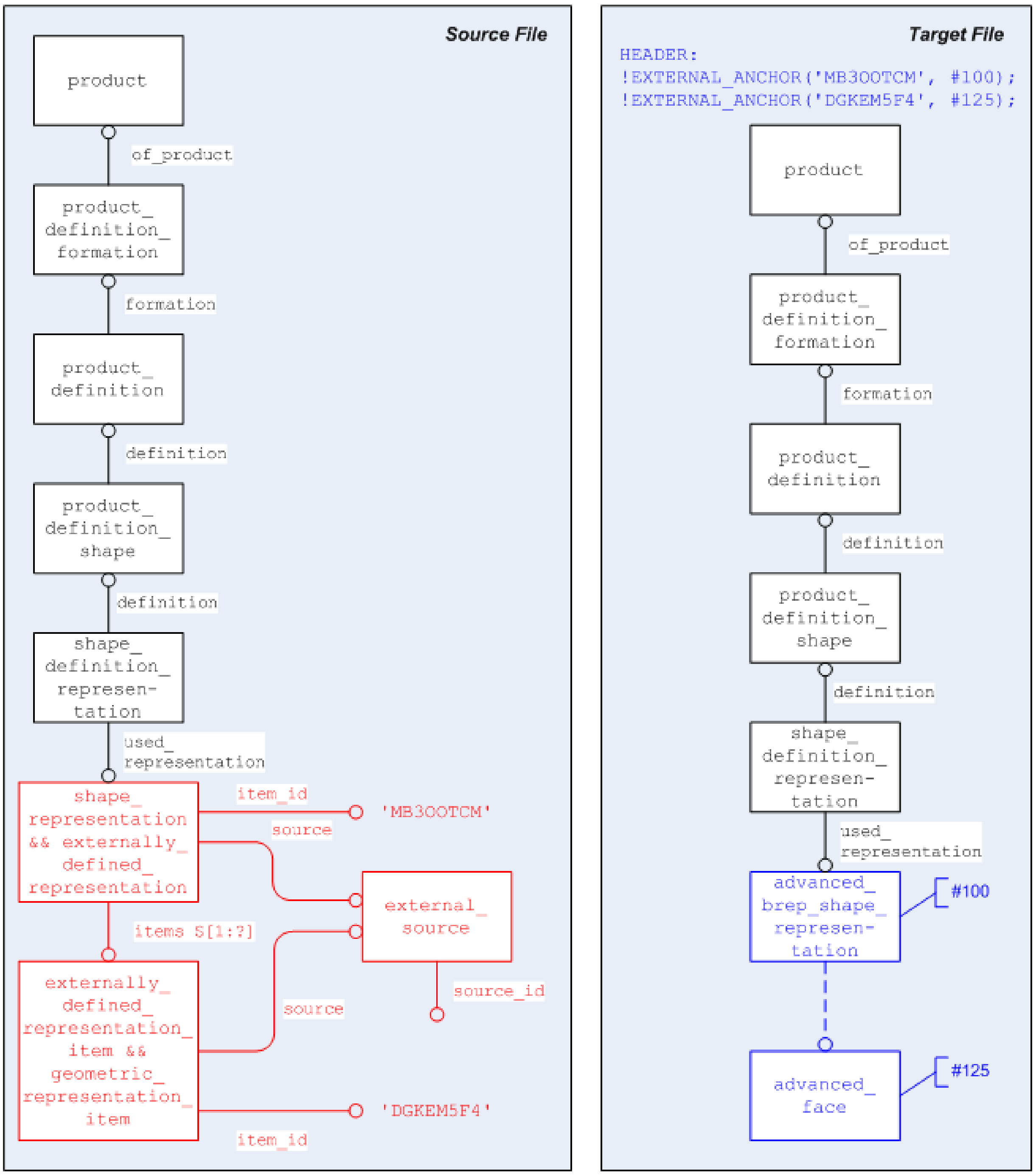
Source File
product
of_product
product_ definition_ formation
formation
product_ definition
definition
product_ definition_ shape
definition
shape_ definition_ represen- tation
used_ representation
shape_ representation && externally_ defined_ representation
item_id
'MB3OOTCM'
source
external_ source
items S[1:?]
source_id
source
externally_ defined_ representation_ item && geometric_ representation_ item
'DGKEM5F4'
item_id
Target File
HEADER:
!EXTERNAL_ANCHOR('MB3OOTCM', #100);
!EXTERNAL_ANCHOR('DGKEM5F4', #125);
product
of_product
product_ definition_ formation
formation
product_ definition
definition
product_ definition_ shape
definition
shape_ definition_ represen- tation
used_ representation
advanced_ brep_shape_ represen- tation
#100
advanced_ face
#125

Figure 24: External Element References to Representation and Representation Item

Note that the complex entities shown in the diagram above are a generic example, and not a recommendation to always use them. The actual use of these entities will be explained in the corresponding domain-specific Recommended Practices.

For all subtypes of `externally_defined_item`, the `item_id` attribute will store the unique ID string that will identify the target instance in the external file via the `EXTERNAL_ANCHOR` header entry. See section 7.2 for notes on its creation.

The `source` attribute will point to an instance of `external_source`. This is in most cases the same instance of `external_source` that is used in the definition of the basic external reference from the source file to the target file, as shown in Figure 17 and Figure 18. Such a basic external reference has to always to be there between files; the EER mechanism extends this capability, but does not replace it. There are, however, some exceptions (in assemblies) where an `external_source` will be used on its own. These cases are described at the respective locations.

7.4 Identification of Assembly Component Instances across Files

One of the main challenges in making EER work is to make it work with nested external references as defined in section 6.4 above. In such a scenario, where the assembly structure is distributed across several files, the path to the correct instance also needs to be identified across several files. This will result in a number of additional external element references, which shall still be manageable.

The previous approach for instance identification across several levels in an assembly was to use instances of `specified_higher_usage_occurrence` (`SHUO`). Each `SHUO` would connect two adjacent levels in the assembly structure, so that in the end a recursive bottom-up structure was created. Especially in the case of larger assemblies, this was perceived to be too complex to handle.

Most modern CAD systems internally handle their data in a different way: If a piece of information is defined in an assembly context, the knowledge about the entire path from this (relative) root node to the involved leaf nodes is stored at the (relative) root node. The intermediate nodes do not "know" that they are being traversed to convey assembly-level data.

7.4.1 Source Files

To reflect this approach, AP242 introduces the new entity type `multi_level_reference_designator` (`MLRD`). Its main attribute is the location, which is an ordered top-down list of `NAUOs`. It defines:

- The <u>root node</u> to be the `relating_product_definition` of the first `NAUO` in the list
- The <u>leaf node</u> to be the `related_product_definition` of the last `NAUO` in the list

Figure 25 below gives an illustration of this straight-forward structure.

As the name suggests, `MLRD` relies on the `NAUO.reference_designator` attribute. Though technically an optional attribute, the following rules apply:

- For any instance of `NAUO` referenced in the location attribute of a `MLRD`, the `reference_designator` attribute must be populated
- The `reference_designator` has to be unique in the context of the `relating_product_definition`.

Reference designators are a construct originating from standards such as IEC 81346-1, where they are defined as "identifier of a specific object formed with respect to the system of which the object is a constituent." They are widely used in the electric domain, and will be used here as well since the existing uniqueness rule on `NAUO.id` is too weak for the given context, and cannot be changed to be more restrictive because this would render existing files invalid.

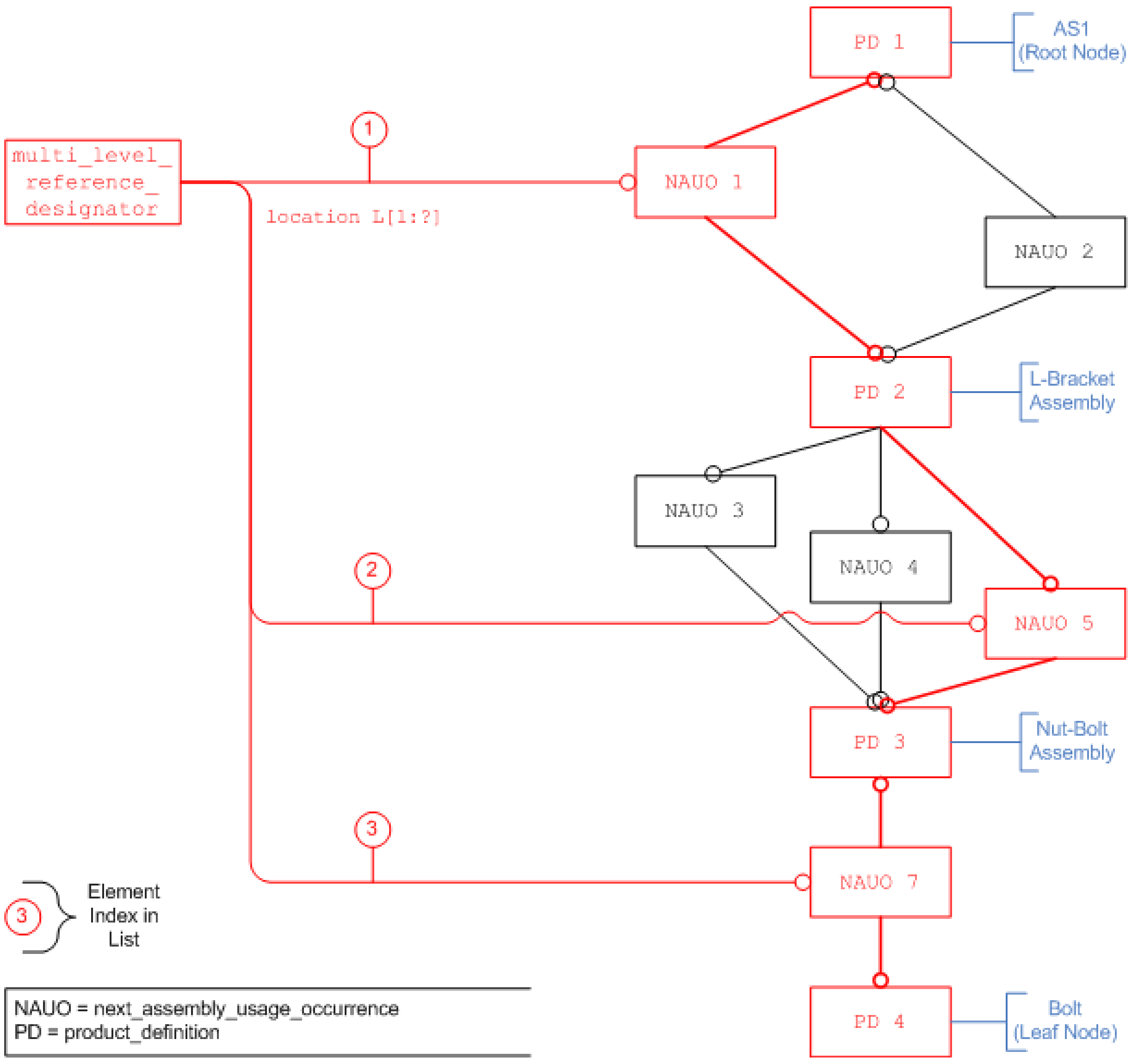

Figure 25: Identification of a Component Instance using MLRD

The key aspects in the creation of an assembly structure are that each node in the assembly tree is represented by a unique trifecta of `product`, `product_definition_formation` and `product_definition`; and that a pair of adjacent `product_definitions` is connected by one or more instances of `NAUO` – as shown above for the three instances of Nut-Bolt Assembly within the L-Bracket Assembly.

So in the context of nested external references, in order to traverse an assembly structure across several files, the following external element references are needed:

- One for each node on the path from the root node to the leaf node
- One for each `NAUO` on the path from the root node to the leaf node

For this purpose, the new entity type `product_definition_reference` has been introduced with AP242, and will be used in combination with `NAUO` as shown below.

Important Note: The following structure for EER in a nested assembly is intended to give "navigation instructions" in an already existing assembly structure. It is not applicable to actually create an assembly structure. This means that the files indicated in Figure 26 below are connected to each other by fully defined external references as defined in section 5 of this document. EER aims to provide additional information, not replace existing data.

As indicated in the Basic Concepts paragraph, the entire knowledge about the path to the desired instance is stored at the root node, here AS1. The dashed green and blue lines in Figure 26 below indicate which entities are linked together via external element references:

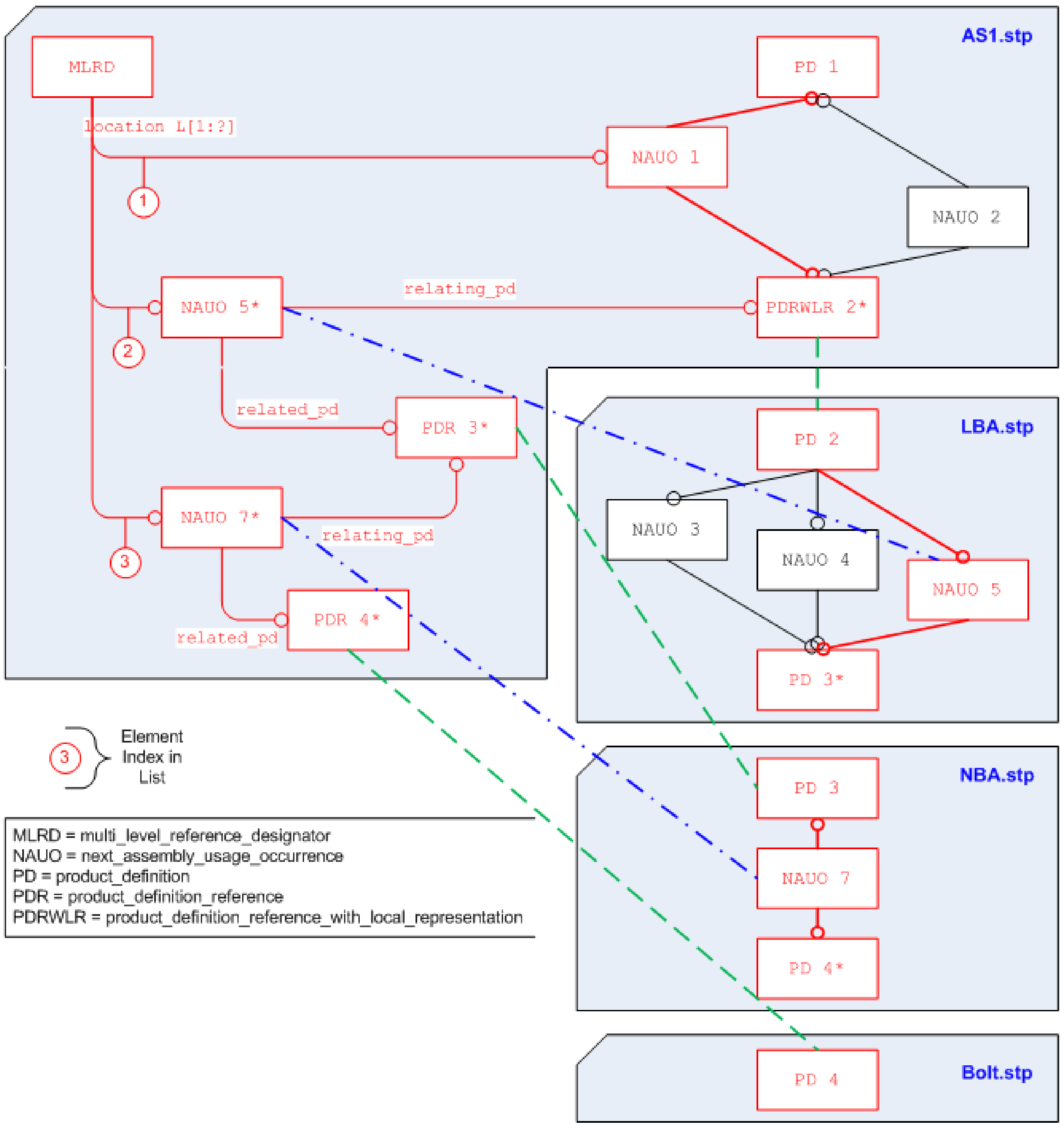

Figure 26: Identification of a Component Instance with Nested External References

For each assembly node defined in an external file, one instance of `product_definition_reference` needs to be created, and its attributes populated in the following way:

Attribute	Population
`source`	`external_source` which holds the name of the target file
`product_id`	Value of the `id` attribute of the targeted `product` (part) in the external file

Attribute	Population
product_definition_formation_id	Value of the id attribute of the targeted product_definition_formation (part version) in the external file
product_definition_id	Value of the id attribute of the targeted product_definition in the external file
id_owning_organization_name	If for the targeted product the "id owner" role is specified (as required in AP214), then this attribute holds the value of the name attribute of the organization assigned to the product via an applied_organization_assignment.

Table 3: product_definition_reference attribute population

Compared to a fully defined external reference as documented in section 5 above, this is a drastically reduced set of information. Nevertheless it is sufficient, since it only defines a reference to information fully defined elsewhere that is also accessible through the usual structure of external references. This approach was chosen to keep EER as lean as possible.

The information given by product_definition_reference is to be interpreted in the following way: In the external file indicated by source, there has to be a product_definition with the given id, that references a product_definition_formation with the given id, which in turn references a product with the given id, that, if specified, has an 'id owner' with the given name. If the file cannot be found, or the pattern cannot be found exactly as described, this is an error and has to be treated as such. Keep in mind that the id_owning_organization_name is optional, but has to be populated if the 'id owner' is defined in the external file.

The blue entities on the left hand side of Figure 27 below illustrate how such a reference works by matching the listed id and name attributes.

There is one special case for the product_definition_reference, and that is the reference to the first externally defined assembly node. For this, there is already a fully defined external reference available, which means that the required key attributes are already defined in the local representation of the externally defined part (left hand side of Figure 17 / Figure 18). Hence it is not necessary to copy these values into the attributes of product_definition_reference, but the product_definition to which the applied_document_reference is attached will be replaced by the subtype product_definition_reference_with_local_representation.

This entity type also has a source attribute, which shall point to the same instance of external_source that is referenced by the applied_external_identification_assignment as shown in Figure 17 and Figure 18. The other key attribute values will be taken directly from the respective entities as illustrated by the red entities on the right hand side of Figure 27 below:

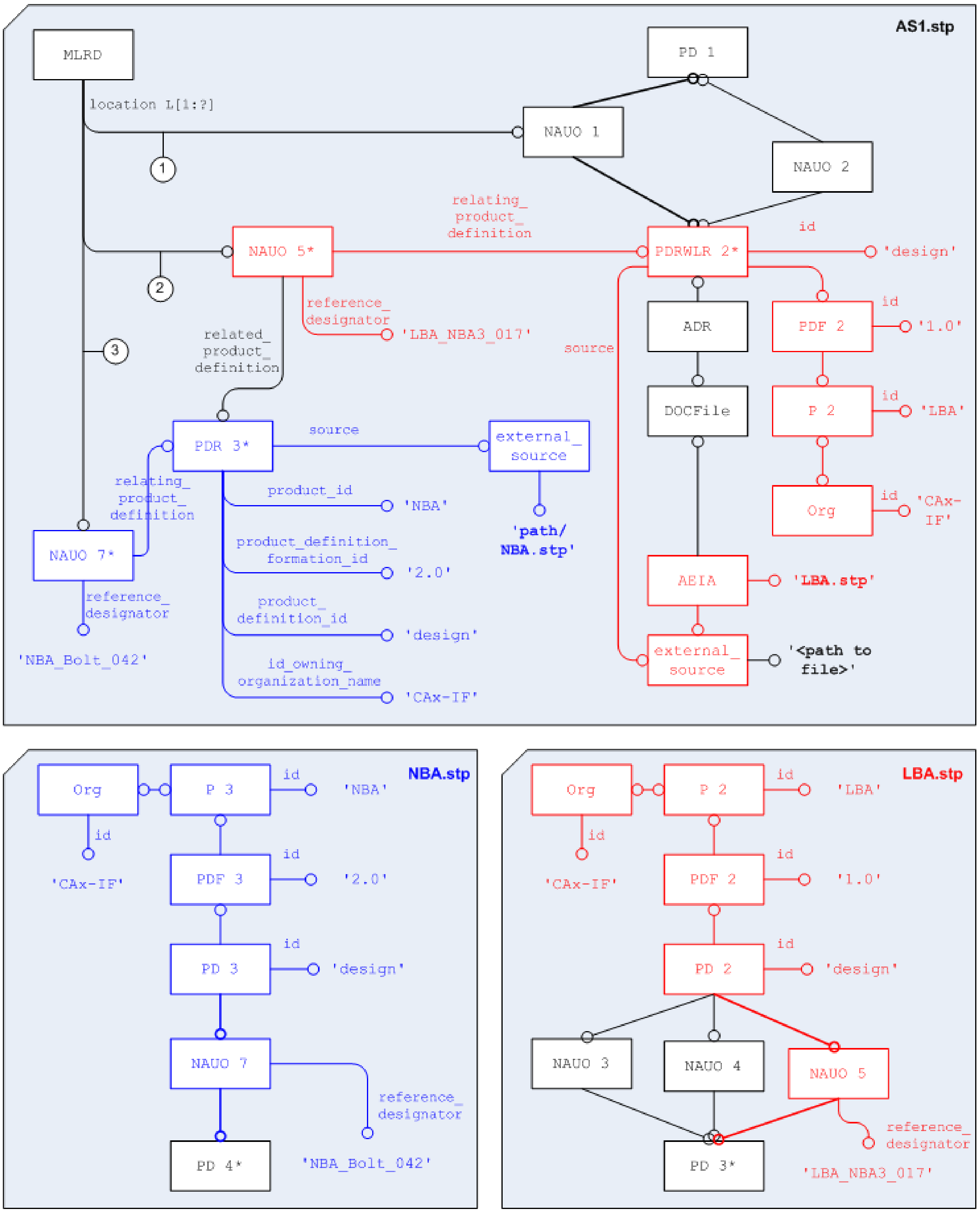

Figure 27: Definition of Key Attributes for Instance Identification in External Files

Now that the references to the externally defined assembly nodes are defined, the correct usages need to be identified for the complete and unambiguous definition of the path. Regardless of whether an assembly node is defined locally or externally, this is done using `NAUO`.

The `multi_level_reference_designator.location` attribute always contains a list of `NAUO`s. For each `NAUO` therein, the following rules apply:

- The `reference_designator` attribute has to be populated, and has to be unique in the context of the `relating_product_definition`.

- If the relating_product_definition is a product_definition, i.e. defined in the same file, then the NAUO is a defining element of the assembly structure.
- If the relating_product_definition is a product_definition_reference (_with_local_representation), i.e. defined in an external file, then the NAUO is to be understood as an external element reference to the NAUO which:
    - can be found in the same file as indicated by the relating product_definition_reference, and
    - which has the same reference_designator attribute value.

Note that since the reference_designator is unique in the context of the relating_product_definition, it is not necessary to define an external element reference to the related_product_definition in the same file.

7.4.2 Target Files

As described above, the externally defined assembly nodes and usages are identified by a (set of) identifying string attribute(s) which allow for unambiguous identification of the target elements. This knowledge is stored entirely at the root node, by the structure reference in the MLRD.location attribute.

This means that in none of the intermediate files that contribute to the definition of the path from root to leaf, any EXTERNAL_ANCHOR header entries are required. As in most native CAD system internal data models, the intermediate nodes do not “know” that they are part of such a path. This also means that additional paths can be defined at a (relative) root node without the need to touch any of the intermediate files.

Product_definition_references, and NAUOs that represent external element references to a NAUO in an external file, are the only types of EER that do not require the definition of EXTERNAL_ANCHOR header entries in the respective target files.

Of course, if the version of a subassembly changes, then the related path information has to be updated accordingly with the appropriate key attribute values. However, this applies to the original external references as well and hence is not a new requirement.

7.5 External Element References to Shape_Aspects

Based on the use cases that were the basis for developing the EER approach, this scenario is currently only supported in assemblies. It will occur when a shape_aspect defined in a component part needs to be used in an assembly context.

7.5.1 Target File

In principle, everything written in section 7.3.1 above applies here as well. For the targeted shape_aspect in the target file, an EXTERNAL_ANCHOR header entry needs to be defined, which translates the given unique ID to the actual entity instance number of the shape_aspect in the data section of the file.

7.5.2 Source File

In the source file, three pieces of information need to be connected in order to define a portion of geometry of an assembly component instance in the context of a higher-level node in the assembly structure. The new entity type component_path_shape_aspect has been introduced to gather this information in the following way:

- The assembly node which defines the context, through the of_shape attribute
- The exact instance of the component, through the location attribute. This can be a single NAUO, or a list of NAUOs defined using MLRD (see above).

- The shape_aspect defined on the component itself, through the component_-shape_aspect attribute.

The following diagram illustrates this:

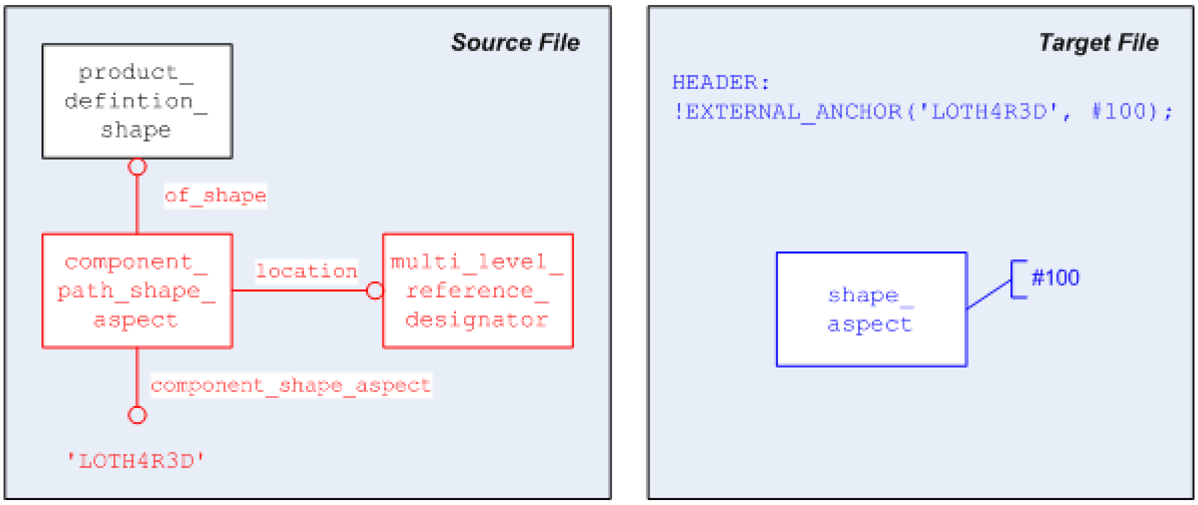

Figure 28: Component_path_shape_aspect

The meaning of this entity can best be explained with a quick example based on the well-known AS1 assembly model (see Figure 22). Let there be a shape_aspect defined in the "bolt" part which describes the head of the bolt. Since it is defined within the component, and instantiated with it, it describes the head of every bolt in the assembly. Using component_path_shape_aspect, it is now possible to define "the head of the bolt in the third Nut-Bolt Assembly within the first L-Bracket Assembly of AS1", and thus assign a specific property to a particular instance.

The definition of the target component instance using the new entity type multi_level_-reference_designator is defined in detail in the previous section. It is basically a list of next_assembly_usage_occurrences that describe a top-down path through the assembly structure. The following rules apply:

- The definition attribute of the product_definition_shape referenced by the component_path_shape_aspect.of_shape attribute has to reference the product_definition for the root node of the path given by the location attribute.
- At the opposite end, the definition attribute of the product_definition_shape referenced by the of_shape attribute of the shape_aspect referenced (directly or via EER) by the component_path_shape_aspect.component_shape_aspect attribute has to reference the product_definition for the leaf node of the path given by the location attribute.
- component_path_shape_aspect can be used in single-file scenarios; in this case, the component_shape_aspect attribute points to the actual shape_aspect instance. If used with external references, the component_shape_aspect attribute contains the unique ID string that identified the shape_aspect in the external component part file.
- component_path_shape_aspect does not have a file reference (external_-source) on its own; it is defined so that the component_shape_aspect will always be found in the same file that contains the original product_definition for the leaf node of the path given by the location attribute.

For the definition of the string identifying the target `shape_aspect` in the external file, please consider sections 7.1 to 7.3 above. For the definition of the `location`, please see section 7.4.

7.6 External Element References to a Generic Chain of Elements

The previous sections gave an overview on the generic EER concepts realized with AP242, which will be detailed in the respective domain and use-case specific Recommended Practices. Based on the use cases discussed thus far, all files involved in an EER scenario are assumed to be STEP files. Hence, external elements to particular entity types such as `representation_item` and `shape_aspect` can be given.

However, the external reference mechanism in STEP is much more powerful and can define references to basically any data format. Consequently, the EER mechanism has been equipped with a generic entity that can store the necessary "navigation instructions" to find a particular data element in such an arbitrary external file, given that the file format provides visible and sufficiently unique element IDs.

The entity is defined as:

```
ENTITY externally_defined_item_with_multiple_references
  SUBTYPE OF (externally_defined_item);
  references : LIST[1:?] OF UNIQUE source_item;
UNIQUE
  UR1: references;
WHERE
  WR1: NOT(SELF\externally_defined_item.item_id IN references);
END_ENTITY;
```

As a subtype of `externally_defined_item`, it gives the `item_id` of the target element in the file indicated by `source`. In addition, it provides a list of unique `source_item` identifiers that are intended to provide a chain of elements leading from the root element of the data structure in the target file to the targeted element. How these identifiers are defined, what they represent, and how they shall be applied depends on the target file format and the underlying use case.

Annex A Part 21 File Examples

STEP files relating to the capabilities described in this document are available in the public STEP File Library on the CAx-IF homepage; see either http://www.cax-if.de/library/ or http://www.cax-if.org/library/.

The files are based on current schemas for both AP203 Edition 2 and AP214, and have been checked for syntax and compliance with the Recommended Practices.

Annex B Availability of implementation schemas

B.1 AP214

The AP214 schemas support the implementation of the capabilities as described. The schemas can be retrieved from:

- IS Version (2001) – http://www.cax-if.de/documents/ap214_is_schema.zip
- 3rd Edition (2010) – http://www.cax-if.de/documents/AP214E3_2010.zip

B.2 AP203 2nd Edition

The long form EXPRESS schema for the second edition of AP203 can be retrieved from:

- http://www.cax-if.de/documents/part403ts_wg3n2635mim_lf.exp

Note that the first edition of AP203 is no longer supported by the Recommended Practices.

B.3 AP242

The capabilities described in this document are also supported by AP242, the joint successor of AP203 and AP214, with no changes to the instantiation.

The latest development longform EXPRESS schema for AP242 can be found in the CAx-IF member area. It will be published on the public web site once approved by ISO.