Recommended Practices

for

PMI Polyline Presentation (AP203/AP214)

Release 2.3

October 13, 2014

Contacts	
Jochen Boy	Phil Rosché
PROSTEP AG	ACCR LLC.
Dolivostraße 11	125 King Charles Circle
64293 Darmstadt / Germany	Summerville, SC 29485 USA
jochen.boy@prostep.com	phil.rosche@accr-llc.com

Table of Contents

List of Figures

List of Tables

Document History

Release	Date	Change
1.0	2008-06-16	Initial release
2.0(*)	2013-05-24	New version based on comprehensive PMI Rec.Pracs. v3.6
2.1(*)	2013-08-20	Integration of agreed PMI Terminology
2.2a(*)	2014-01-31	Addition of Combination of Validation Properties (6.3.5), Notes concerning shape_aspect.id and supplemental geometry.
2.3	2014-10-13	Editorial changes for publication

(*): Internal review versions; not published.

1 Introduction

Product and Manufacturing Information (PMI) is one of the major building blocks for the 3D model-based enterprise. The goal is to eventually replace all 2D drawings with 3D models that are enriched with all necessary data to fulfill the respective use case.

There are two main approaches for providing PMI data in a model:

- Semantic Representation – the machine-consumable, associative, intelligent storage of PMI definitions in the model so that they can be used in downstream applications or to edit the model.
- Presentation – the human-readable display of the PMI data in the 3D model in an organized way. The presentation can occur as graphics or character-based.

Figure 1 below illustrates how these approaches are related to each other. Please refer to Annex B for a full definition of terms.

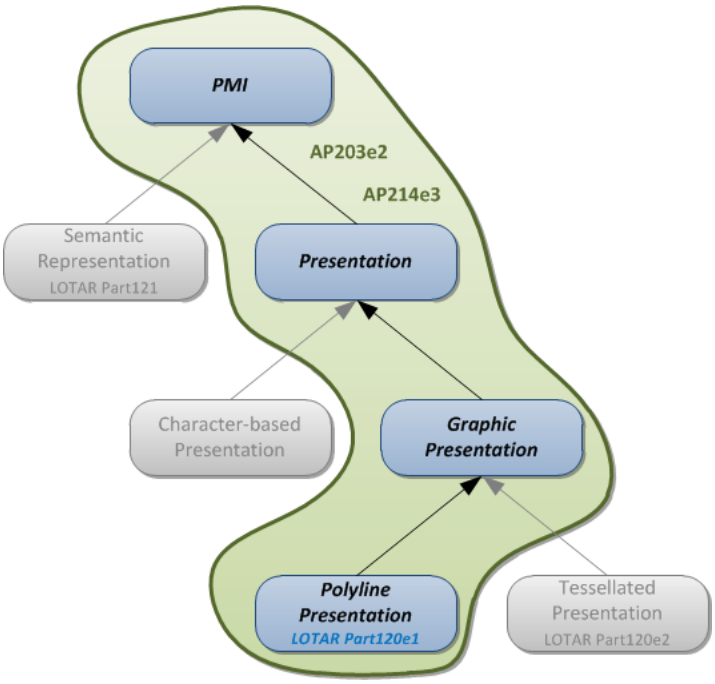


Figure 1: PMI Approaches and Scope of this Document

This document deals with the presentation of PMI elements and annotations in graphical form, more specifically with the "Polyline Presentation" approach as supported by STEP AP203 2nd Edition (2011) and AP214 3rd Edition (2010).

This means that the information to be displayed is broken down into geometric elements, namely line segments and circular arcs. The advantages of Graphic Presentation are that since it is rendered, it is presented in the STEP file exactly as it was in the CAD system. Since the instantiation is done using basic geometric entities, generally every viewer or CAD system which finds the link in the file structure can display it.

The original "Polyline Presentation" approach based on AP203e2 and AP214e3 as described in this document has some limitations, which will be clearly pointed out. These limitations have been resolved in the new STEP AP242, the joint successor of AP203 and AP242. The implementation of Graphic Presentation based on AP242, as well as the implementation of Semantic PMI Representation, is described in the "CAx-IF Recommended Practices for the Representation and Presentation of Product Manufacturing Information (PMI)", version 3.6 or later.

2 Scope

The following are within the scope of this document:

- The definition of PMI Polyline Presentation based on AP203e2 and AP214e3
- The organization of PMI Annotations using Saved Views
- The definition of Validation Properties for PMI Polyline Presentation

The following are out of scope for this document:

- The definition of PMI Graphic Presentation based on AP242
- The definition of Semantic PMI Representation

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.---PMI Polyline Presentation---2.3---2014-10-13

It will appear in a STEP file as follows:

```
FILE_DESCRIPTION(('...','CAx-IF Rec.Pracs.---PMI Polyline Presentation---
2.3---2014-10-13',),'2;1');
```

4 Fundamental Concepts

The information to be presented can be converted to geometry in different ways, which have an impact on how they will become instantiated in the STEP file. The main characteristics to be considered are:

- Outline Characters: Elements are shown by their outline
- Filled Characters: Elements are shown as solid or filled
- Stroked Characters: Elements are defined by their centerline
- Geometric Elements: These include leader and witness lines, bounding boxes of feature control frames, and arrows (which may be filled).

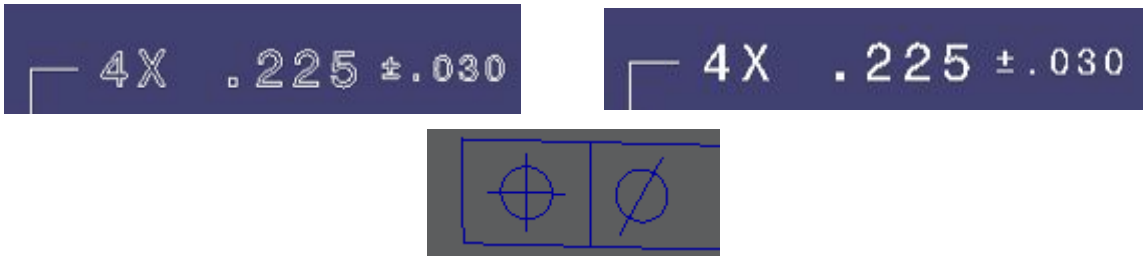


Figure 2: Outline Characters (upper left), Filled Characters (upper right), Stroked Characters and Geometric Elements (below)

Independent from this, Graphic Presentation elements can be styled and arranged in Saved Views (see section 5.3.2) to provide the relevant set of PMI data to the user in the format defined on export. The definition of various validation properties (see section 1) help to ensure the correctness and completeness of the information displayed.

4.1 Polyline Presentation

A polyline is a line created by a series of short straight line segments. The entity `polyline` is defined in Part 42, and already known in the AP214 / AP203 Ed2 standards.

Using this type of entity, each PMI feature and 3D annotation can be exported as a `geometric_curve_set` of `polylines`, `circles` and `trimmed_curves` where the basis curve is a `circle` (circular arcs). Polylines are defined by a list of `cartesian_points`. The use of `composite_curves` is also allowed; see section 0 below for details.

Note that these shall be located in a plane parallel to the definition plane of the `annotation_-plane` the PMI element is assigned to, as described in section 5.1 below. Each Polyline annotation has to be in a set of elements of an `annotation_plane`.

4.1.1 Basic Polylines

The following figure illustrates the basic structure for a PMI element presented by Polylines. This can be used to handle stroked and outline characters, as well as geometric elements:

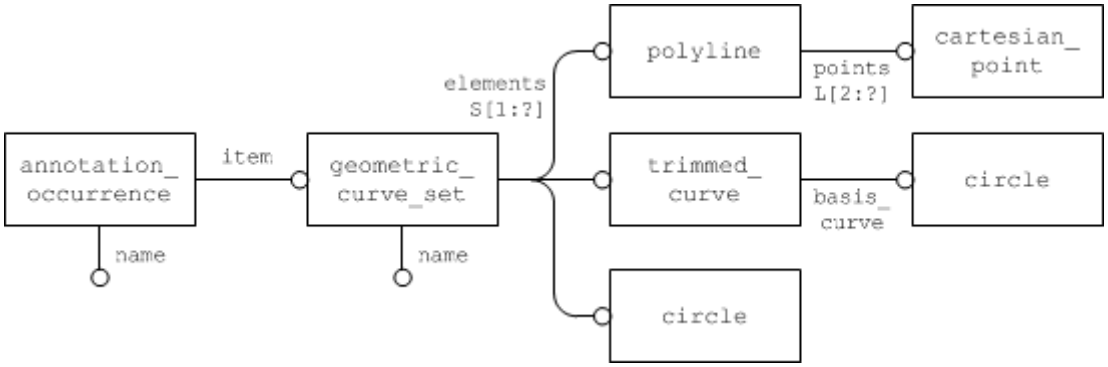


Figure 3: Basic Polyline Definition

Part21 Example

```
#581=ANNOTATION_OCCURRENCE('Simple Datum.1',(#580),#577);
#577=GEOMETRIC_CURVE_SET('datum',(#582,#592,#597,#600,#606));
#582=POLYLINE('Simple
Datum.1',(#583,#584,#585,#586,#587,#588,#589,#590, #591));
#592=POLYLINE('Simple Datum.1',(#593,#594,#595,#596));
#597=POLYLINE('Simple Datum.1',(#598,#599));
#600=POLYLINE('Simple Datum.1',(#601,#602,#603,#604,#605));
#606=POLYLINE('Simple Datum.1',(#607,#608,#609,#610));
```

Compatibility Notes

- In AP203e2 and AP214e3, the “anchor” entity for a basic polyline annotation is a plain `annotation_occurrence`.
- In AP242, the subtype `annotation_curve_occurrence` is used with the addition of a `draughting_callout` pointing to it. Refer to the comprehensive PMI Recommended Practices (v3.6 or later) for details.

Pre-Processor Recommendations

The following name attributes are of relevance:

- `annotation_occurrence.name` carries the "user name" of the annotation, in order to maintain compatibility with implementations not using the `draughting_callout`.
- `geometric_curve_set.name` indicates the presented type of PMI; see section 4.3 below for details.

4.1.2 Filled Polylines

The next figure shows the definition of a set of filled polylines. Note that the geometric definition of the boundaries is the same as above. The specific subtype of `annotation_occurrence` carries the information that these characters are to be filled.

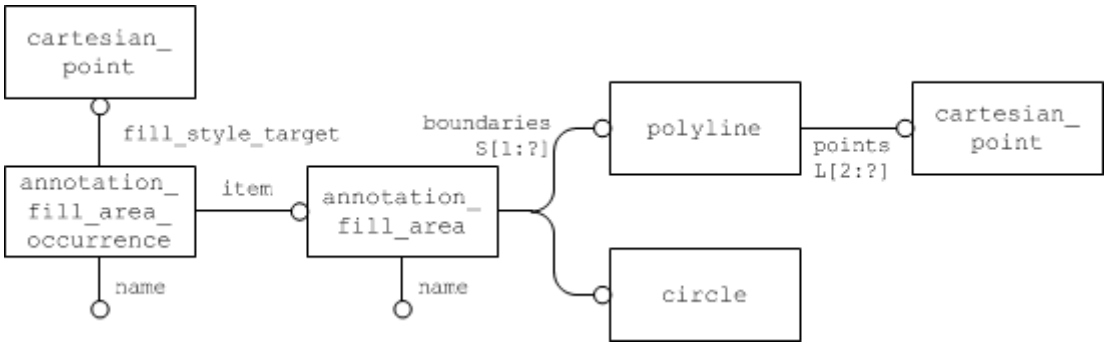


Figure 4: Filled Polyline Definition

Pre-Processor Recommendations

The curves referenced as boundaries of the `annotation_fill_area` need to fulfill the following requirements:

- Each boundary has to be a closed curve
- No two of these closed curves may intersect
- All boundaries have to be coplanar.

It is allowed to have several outer boundaries in one set, so an entire text string can be presented with one `annotation_fill_area`. One "inner point" needs to be defined to indicate where the filling shall start. Even though efficient algorithms are available to determine the filled and unfilled portions of the annotation, the following convention can help to simplify the process, if the STEP processor is capable of manipulating the `polylines` this way:

- Outer boundaries (enclosing a filled area) shall be oriented in a positive sense, i.e. counter-clockwise (by giving the points defining the polyline in the appropriate order)
- Inner boundaries (enclosing an unfilled area) shall be oriented the opposite way, i.e. clockwise.

Note that due to the various algorithms that exist to determine the inner and outer boundaries in a set of closed curves like this, geometrically it would be most efficient to have only one outer boundary per `annotation_fill_area`. But that would inflate the STEP file structure and make evaluation and specifically validation much more complicated, if not impossible. It is therefore strongly recommended to not split up annotations below the granularity corresponding to the "Equivalent Unicode String" (see section 6.3.3). For instance, all boundaries belonging to the presentation of the real value "3.200" have to be in the same `annotation_fill_area`.

The use of Composite Curves

Especially for the support of Filled Polylines, it can be meaningful to group geometric elements together in order to describe more complex boundaries consisting of Polylines as well as trimmed curves (circular arcs). This allows the rendering of complex characters in a more exact way, e.g. to construct the shape of a 'D' from a polyline and a half circle.

Hence, each boundary can be described as a `composite_curve`. Each portion of the boundary is declared as a `composite_curve_segment`, and the `transition` attribute helps to satisfy the requirements for the boundaries of an `annotation_fill_area` as listed above.

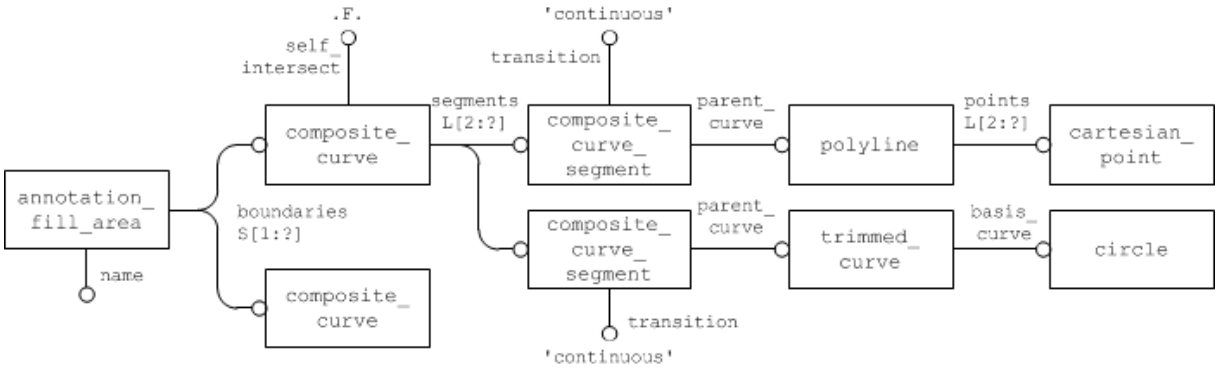


Figure 5: Definition of boundaries using composite_curves

Concerning the attribute values of `composite_curve` and `composite_curve_segment`, the following recommendations are given:

- `composite_curve.self_intersect` strictly has to be `FALSE` for the boundaries of Filled Polylines
- `composite_curve_segment.transition` shall always be `'continuous'` with one exception: it may be `'discontinuous'` for the last segment of a `composite_curve` to indicate it is open, e.g. when using it for stroked font or geometric elements in an annotation.
- `composite_curve_segment.same_sense` can be used to manipulate the boundary orientation, hence indicating inner and outer boundaries as described above.

Post-Processor Recommendations

In case an importing STEP processor capable of handling only basic polylines (see 4.1.1) encounters a file with filled polylines (as defined in 4.1.2), these can still be imported by re-interpreting them in the following way:

- Treat the `annotation_fill_area_occurrence` as if it were an `annotation_occurrence`. The `cartesian_point` given as fill style target as well as any `fill_area_styles` shall be ignored. `curve_styles`, however, can be processed as usual.
- Treat the `annotation_fill_area` as if it were a `geometric_curve_set`.

4.2 Graphic Annotation Subsets

There are quite a number of reasons why a presentation element that is a single annotation in the source system may get split into several annotations during export to STEP. With Polylines, it is clear that stroked/outlined elements have to be in a separate annotations from filled elements due to the different required entity types (see 4.1.1 and 4.1.2), or a processor may decide to distinguish between textual and graphical elements of an annotation.

Basically all kinds of graphic presentation can be combined into a single annotation; it is reasonable to have the filled characters of a text combined with basic polylines for the leader lines and other geometric elements.

Pre-Processor Recommendations

For all subsets defining one PMI element, the following rules apply:

- The `name`s of the `annotation_occurrence` and `annotation_fill_area_occurrence` entities have to be identical. This attribute carries the name of the PMI element as defined by the user or CAD system in the native model.
- The names of the `geometric_curve_set` and `annotation_fill_area` entities have to be identical. This attribute carries basic information about the type of PMI (see section 4.3 below).
- The `annotation_(fill_area_)occurrences` have to be linked together by `annotation_occurrence_relationship` entities with a name of 'associated curves'. The relationship shall be established from (`relating_annotation_occurrence`) to (`related_annotation_occurrence`) in the order:

Filled Polylines → Basic Polylines (Stroked/Outline Characters) → Basic Polylines (Geometric Elements)

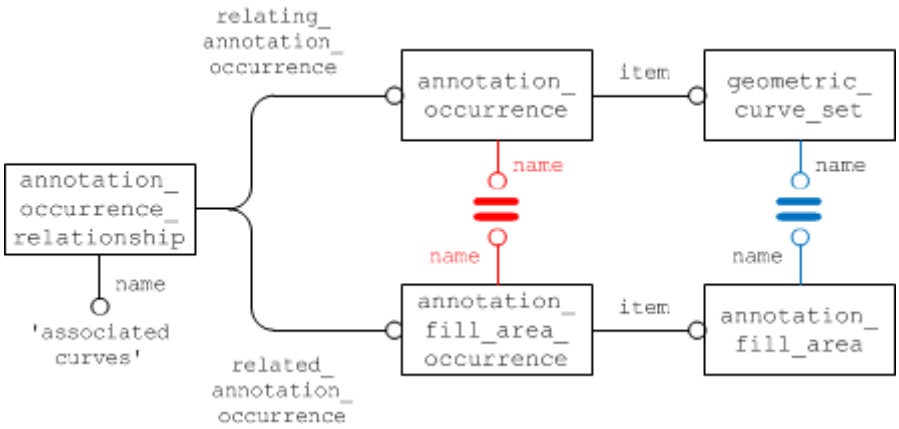


Figure 6: A PMI element defined as a combination of Filled and Basic Polylines

Compatibility Notes

- In AP203e2 and AP214e3, the subsets of annotation will be linked together as described above.
- In AP242, all `annotation_occurrence` subtype instances belonging to the same annotation (PMI element) will be in the `contents` set of the same `draughting_callout`. The identical naming rules for the 'user name' and the indicated PMI type still apply.

Post-Processor Recommendations

When importing an annotation comprised of several subsets, this shall be done so that in the target system, the result for the user is again a single element in the model tree.

4.3 Indicating the Presented PMI Type

It is recommended that the name of the `geometric_curve_set` (for Basic Polylines), or the `annotation_fill_area` (for Filled Polylines) to be included in the STEP file is taken from the list given below. These are based on the ANSI/ISO standards NF EN ISO 1101. The

intention is to provide the user with a harmonized list of names for display in the feature tree or as a property.

Note that the name of the `geometric_curve_set` / `annotation_fill_area` is not intended to transport any intelligent (semantic) PMI information.

Tolerance Types	Dimension Types	Datum Types	Other
angularity	linear dimension	datum	note
circular runout	radial dimension	datum target	label
circularity	diameter dimension		surface roughness
coaxiality	angular dimension		weld symbol
concentricity	ordinate dimension		
cylindricity	curve dimension		
flatness	general dimension		
parallelism			
perpendicularity			
position			
profile of line			
profile of surface			
roundness			
straightness			
symmetry			
total runout			
general tolerance			

Table 1: Suggested list of names

4.4 Styling the Annotation

Each annotation transformed into Polyline Presentation form shall preserve its graphic characteristics (color, line type and width) and optional attributes (type of annotation, layer). The graphical attribute can be global for the annotation.

The style for the presentation will be defined at the applicable `annotation_occurrence` subtype, which in turn is a subtype of `styled_item`. The styles defined at this level shall be applied to all entities in the annotation.

Basically, all definitions given for the styling of surfaces and curves provided in the “Recommended Practices for Model Styling and Organization” apply to Polyline Presentation as well. This also means that each annotation shall have at least one style assigned, either directly as described here or indirectly via a `draughting_model` it is contained in (see section 5 below).

Note that in the case of Filled Polylines, a `fill_area_style` as well as a `curve_style` may be defined. In order to guarantee interoperability with systems not supporting filled characters, the `curve_style` shall always be given if the presentation is styled.

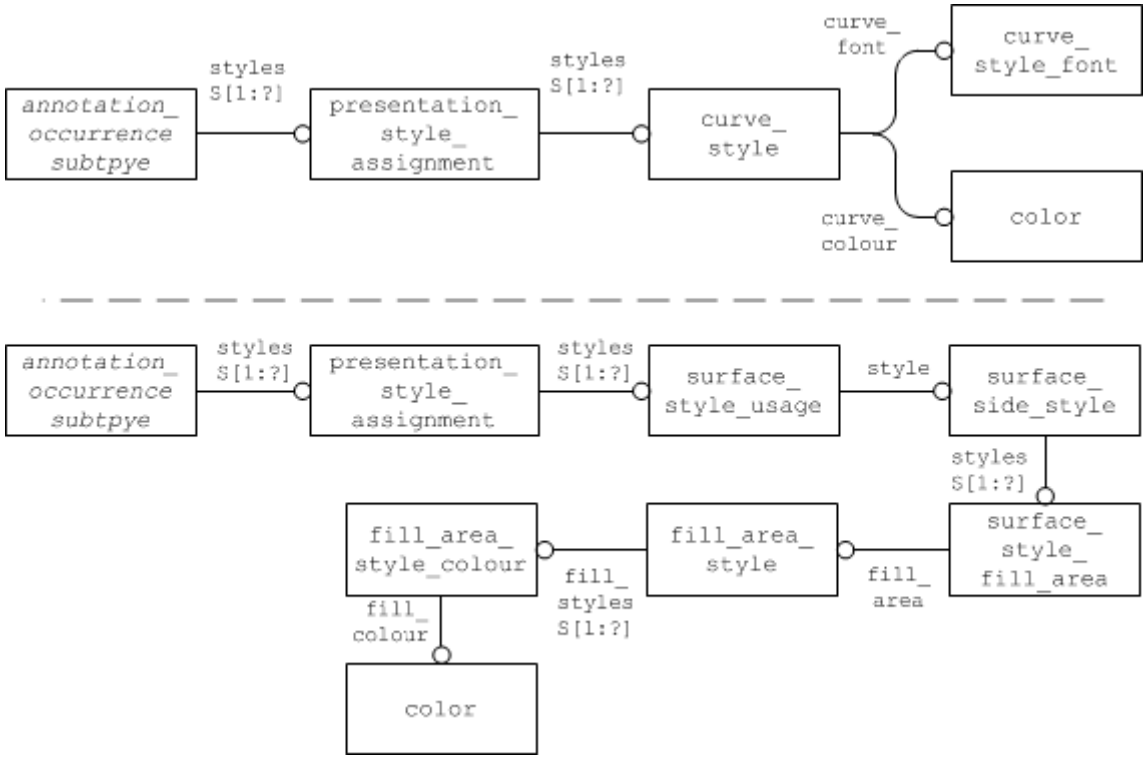


Figure 7: Styling Polyline Annotations: outline/stroked (top) and filled (bottom) Elements

If certain elements within a graphically presented annotation shall have a different style (e.g. the text shall have a different color than the frame), this will be applied through a complex entity composed of `over_riding_styled_item` and the applicable `annotation_occurrence` subtype:

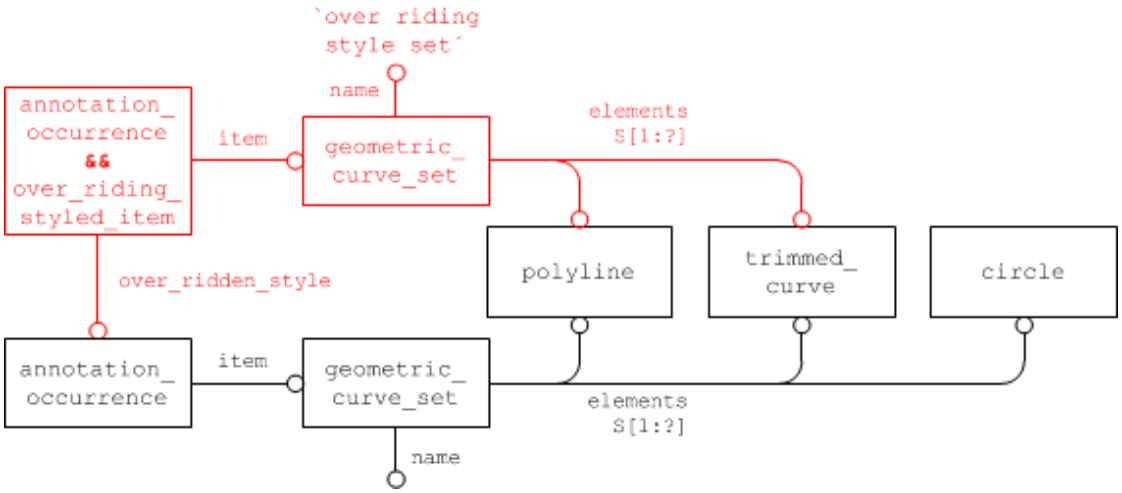


Figure 8: Definition of the overriding style for a Polyline annotation

Figure 8 illustrates this for a Basic Polyline. The concepts described in the “Recommended Practices for Model Styling and Organization” for overriding styles apply here as well. Since the portion of the presentation to be re-styled will in general consist of many constituents, an additional geometric set is needed to collect these. This shall reference the relevant subset of the geometric elements that are contained within the full set of presentation data for this PMI element, and have the name “over riding style set” so that it is clear that it is not a PMI element by itself (refer to section 4.3 above).

5 Presentation Organization

Regardless of the presentation approach used (Polyline / Tessellated / Semantic), the presented data can be organized in various ways. The main approaches for this are the use of annotation planes, saved views, and the capability to link annotations with the associated geometry they provide information about.

5.1 Annotation Planes

In order to position PMI data and 3D annotations on the screen, users usually work with annotation planes. Each graphic annotation has to be assigned to a reference plane and positioned parallel to that, at a specific position related to the geometry. In some systems, the assignment of an annotation to an annotation plane also has an organizational aspect in addition to positioning the annotation in 3D space.

This position must be preserved after conversion of PMI and 3D annotations. In the case of Polyline Presentation, the three-dimensional `cartesian_points` defining the `polylines` shall be located in a plane that is parallel to the definition plane of the `annotation_plane`.

Note that the orientation of the `annotation_plane` should be chosen so that it also indicates the reading direction of the annotations assigned to that plane, i.e. the underlying axis placement shall be defined so, that

- The x axis is the “line that is being written on”, indicating the reading direction
- The y axis points upward from the base to the top of the characters
- The z axis points towards the reader

The same applies for the measuring direction of dimensions given as Graphic Presentation – this should also be taken from the orientation of the `annotation_plane`, if not from the order of related geometric elements.

Note that the definition plane of the `annotation_plane` can be given in two different ways: as (infinite) `plane`, defined by an axis placement, or as `planar_box`, defined by an axis placement, a size in x and a size in y (relative to the axis placement). Each pre-processor may choose which definition fits its internal data structure best; both definitions shall be supported on import.

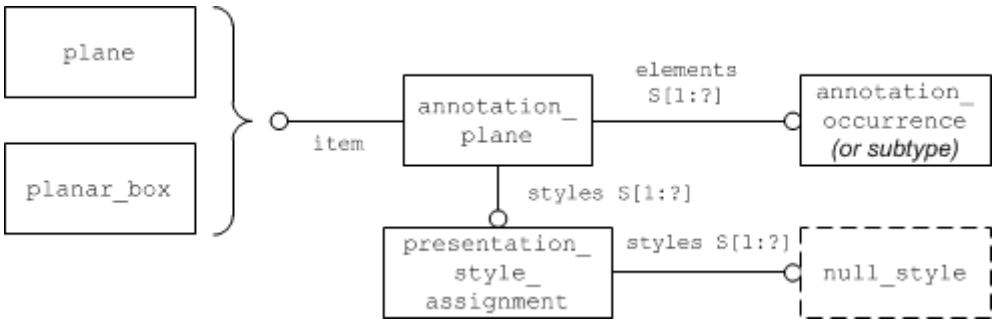


Figure 9: Definition of the Annotation Plane

Part21 Example

```
#571=PLANE('Front View.1',#570);
#572=ANNOTATION_PLANE('Front View.1',(#573),#571,(#581,#614));
#573=PRESENTATION_STYLE_ASSIGNMENT((NULL_STYLE(.NULL.)));
#581=ANNOTATION_OCCURRENCE('Simple Datum.1',(#582),#577);
#614=ANNOTATION_OCCURRENCE('Circularity.1',(#615),#611);
```

Note that due to the way annotations are created and handled in many CAD systems, the elements describing an annotation have to be in a plane parallel to the plane defining the `annotation_plane`, and not necessarily exactly on that plane.

5.2 Global Draughting Model

In order to correctly include all the annotations in the STEP file structure so that they can be easily found and organized later on, they will be collected in one `draughting_model`. Since this `draughting_model` relates to all annotations in the file, it is called the "global" `draughting_model`.

It references all `annotation_planes` in the file, which, in turn, include all `annotation_occurrence` subtypes in their sets of elements:

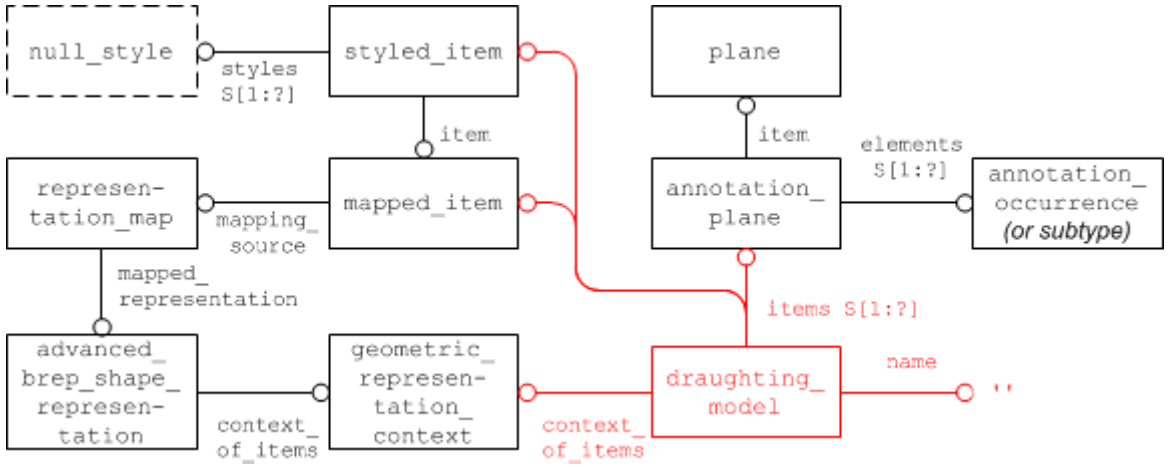


Figure 10: Linking the Annotations together

This global `draughting_model` also contains references to all geometry that shall be displayed when viewing the model. Details about the application and interpretation of styles are described in the "Recommended Practices for Model Styling and Organization", section 4.1 ("Global Styling Container"), and also in this document, section 5.3.2 ("Saved Views").

5.3 Linking Annotations with other Elements

Though annotations by themselves already provide a lot of valuable information to the consumer of a model, their usefulness can be vastly enhanced by putting them in context, i.e. linking them with other elements in the file.

In particular, two "targets" are of interest:

1. Linking Annotations with Geometry ("cross-highlighting")
2. Linking Annotations to other Annotations (grouping)

Note that in the scope of AP242, a third "target" is the Linking of Annotations to Semantic PMI Representation data.

5.3.1 Linking Annotations with Geometry ("Cross-Highlighting")

3D annotations in general and PMI elements in particular are usually linked with geometry, i.e. a specific portion of the geometric shape. For the user, this is typically evident by the fact when a face or edge on the model is selected (highlighted), all associated annotations become highlighted as well, and vice versa. This capability hence is referred to as "cross-highlighting".

This requires two steps: first, the geometry needs to be identified; then, the annotation will be linked to it.

Identifying the Geometry

In order to define the portion of the geometry the annotation relates to, first the corresponding geometric elements have to be identified. In the example in Figure 11 below, this is an `avanced_face` linked through some chain of elements to the `advanced_brep_shape_representation` defining the geometric shape. Next, a `shape_aspect` will be defined so that the face can be referred to. The link between the `shape_aspect` and the `advanced_face` is created via an entity of type `geometric_item_specific_usage`:

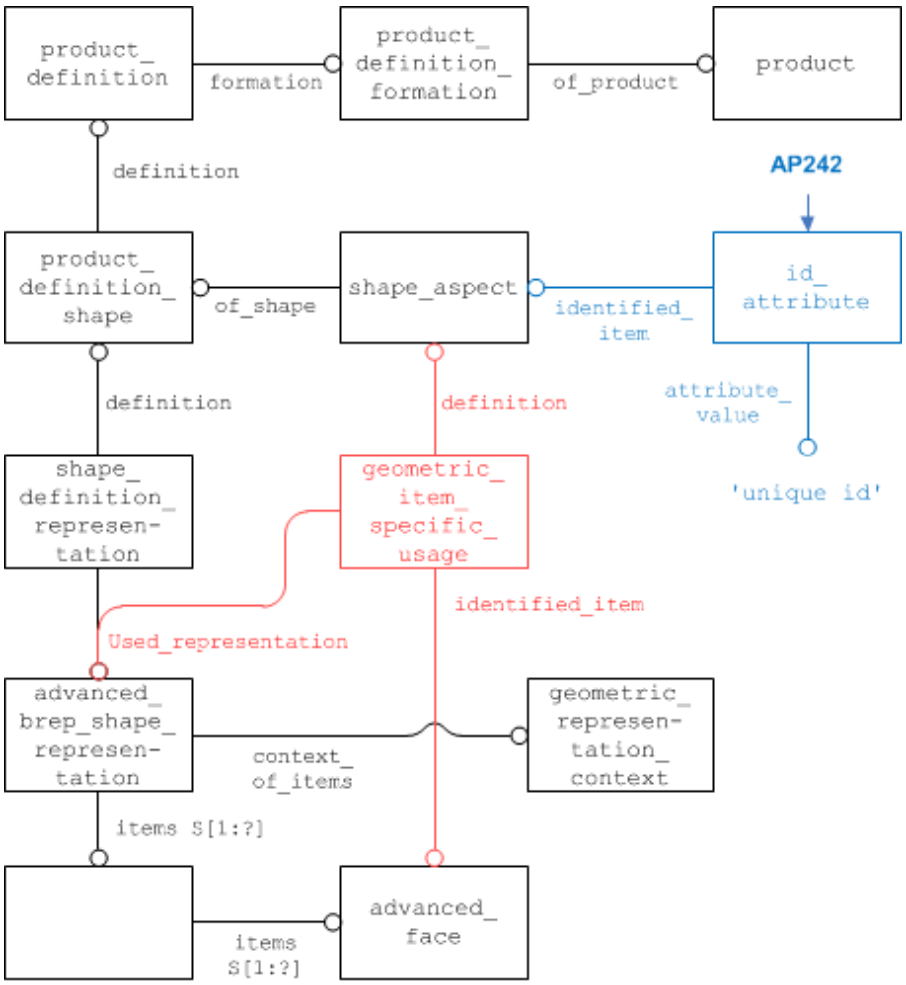


Figure 11: Identification of the relevant portion of Geometry

Part21 Example

```
#15=PRODUCT_DEFINITION_SHAPE('','',#14);
#21=(GEOMETRIC_REPRESENTATION_CONTEXT(3)GLOBAL_UNCERTAINTY_ASSI
GNED_CONTEXT((#20))GLOBAL_UNIT_ASSIGNED_CONTEXT((#16,#17,#19))R
EPRESENTATION_CONTEXT('',''));
#24=ADVANCED_BREP_SHAPE_REPRESENTATION('NONE',(#26),#21);
#25=SHAPE_DEFINITION_REPRESENTATION(#15,#24);
#26=MANIFOLD_SOLID_BREP('PartBody',#35);
#35=CLOSED_SHELL('Closed Shell',(#75,#124,#136,#180,#211,#266,
#290,#350,#410,#432,#439,#461,#468,#492,#509,#526,#543));
#350=ADVANCED_FACE('PartBody',(#313,#331,#349),#295,.T.);
#1180=SHAPE_ASPECT('','GDT',#15,.F.);
#1181=ID_ATTRIBUTE('PartBody for Simple Datum.1',#1180);
#1182=GEOMETRIC_ITEM_SPECIFIC_USAGE('','GDT',#1180,#24,#350);
```

Shape Aspect Identification in AP242

In AP242, there is a uniqueness rule on each of `shape_aspect`, `dimensional_location`, `dimensional_size` and `shape_aspect_relationship` which requires the attribute pair (`id`, `of_shape`) to be unique if the ID attribute exists. There is also a global rule requiring uniqueness of the ID attribute across population of a collection of the above entity types if the ID attributes exist. These rules have been introduced in the context of the Semantic Product and Manufacturing Information (PMI) Representation capabilities and External Element References (EER). The second rule is more restrictive as it requires coordination amongst several entity types. For backward compatibility reasons, AP242 does not require the ID attribute to exist.

Since the ID attribute is derived, an instance of `id_attribute` must be populated, which has the ID string as its `attibute_value` and any of the aforementioned entity types as `identified_item`.

While adding the `id_attribute` is allowed but not required in the formal AP242 document, omitting it in an AP242 file will violate the business agreement for Semantic PMI and EER. Also, in order not to have to make the decision what purpose a `shape_aspect` is used for, it is recommended to add an `id_attribute` to all instances of the above entities, with an `attribute_value` string that is unique among all instances of `id_attribute` in the context of the respective `product_definition_shape`, i.e. if there are 8 `id_attributes` that reference a combination of the above types which all reference the same `product_definition_shape` in their `of_shape` attribute, there shall be 8 distinct values of `attribute_value`.

There is no business requirement to add `id_attribute` in AP203e2 or AP214 files, since Semantic PMI and EER are out of scope for these APs. It is, however, technically legal to do so.

Linking the Annotation

The next step is to link the `draughting_callout` for the annotation to the identified geometry. This is done using a `draughting_model_item_association`, which references the "global" `draughting_model`, the identified item and the `shape_aspect` that relates to the portion of the geometry that the annotation relates to, as defined above.

Figure 12 illustrates the complete structure to identify the relevant portion of geometry and link the annotation to it. The entire path for the link is highlighted.

Part21 Example

```
#566=DRAUGHTING_MODEL('',(#572,#887),#21);
#572=ANNOTATION_PLANE('Front View.1',(#573),#571,(#581,#614));
#581=ANNOTATION_OCCURRENCE('Simple Datum.1',(#582),#577);
#614=ANNOTATION_OCCURRENCE('Circularity.1',(#615),#611);
#1180=SHAPE_ASPECT('','GDT',#15,.F.);
#1182=DRAUGHTING_MODEL_ITEM_ASSOCIATION('','',#1180,#566,#580);
```

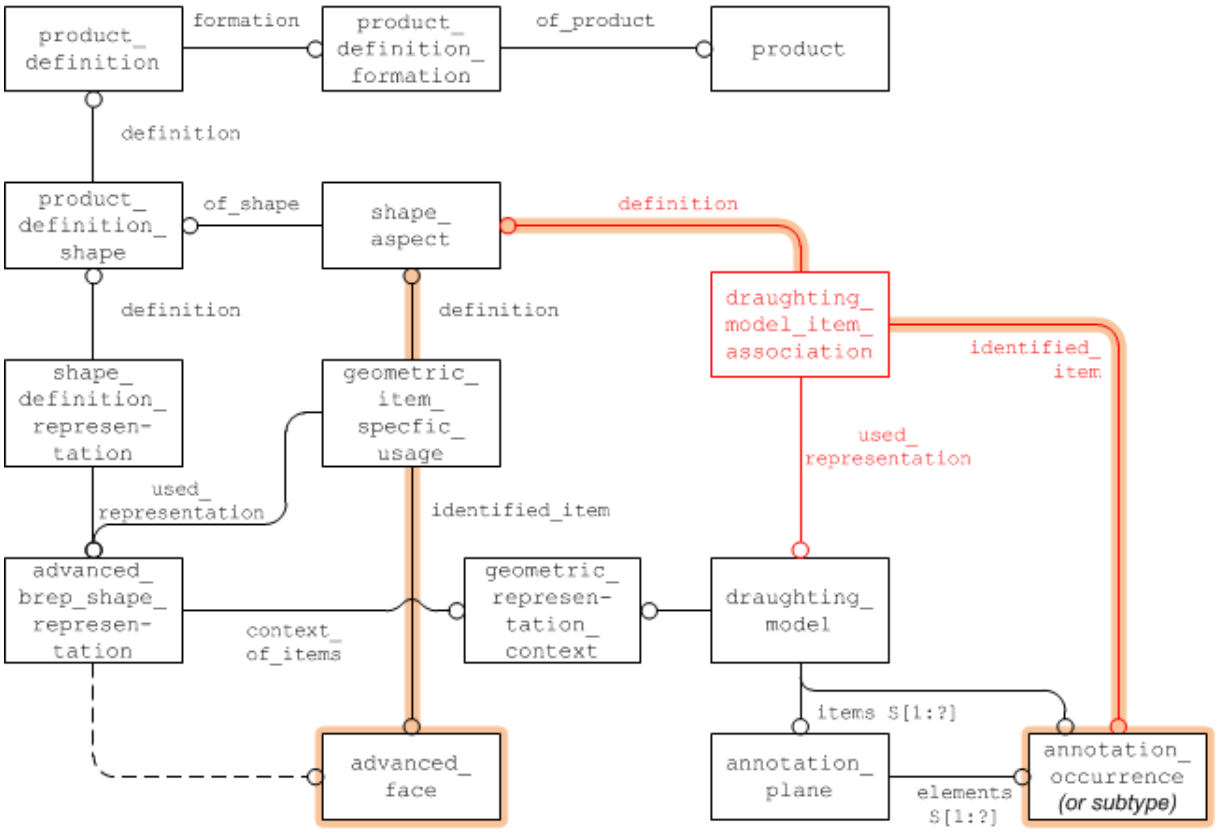


Figure 12: Associating the Annotation with the Geometry

5.3.2 Linking Annotations to other Annotations

There are scenarios where it is useful to put annotations in a relationship with each other. For instance, for the “cross-highlighting” capability described above, it shall be possible to link an annotation presenting a datum reference to the annotation presenting the datum.

Within the scope of AP203e2 and AP214e3, this could be done using instances of `annotation_occurrence_relationship` with a pre-defined magic string. However, this entity type is already used to group subsets of annotations together, which is more important. The CAx-IF decided that using `annotation_occurrence_relationship` also for linking annotations to other annotations introduces a too high risk of ambiguity, hence this capability is not supported in AP203e2 and AP214e3.

Note that in AP242, where `draughting_callout` is the “anchor” entity for an annotation, this capability is supported using `draughting_callout_relationship` (see version 3.6 or later of the comprehensive PMI Rec. Pracs.).

5.3.3 Grouping Annotations with Annotation Planes

As described in section 5.1 above, each graphic presentation annotation is bound to one `annotation_plane`. The primary objective of this relationship is to provide information about location and orientation of the annotation. Some CAx systems also interpret annotation entities which are defined on the same `annotation_plane` as a means of grouping the data.

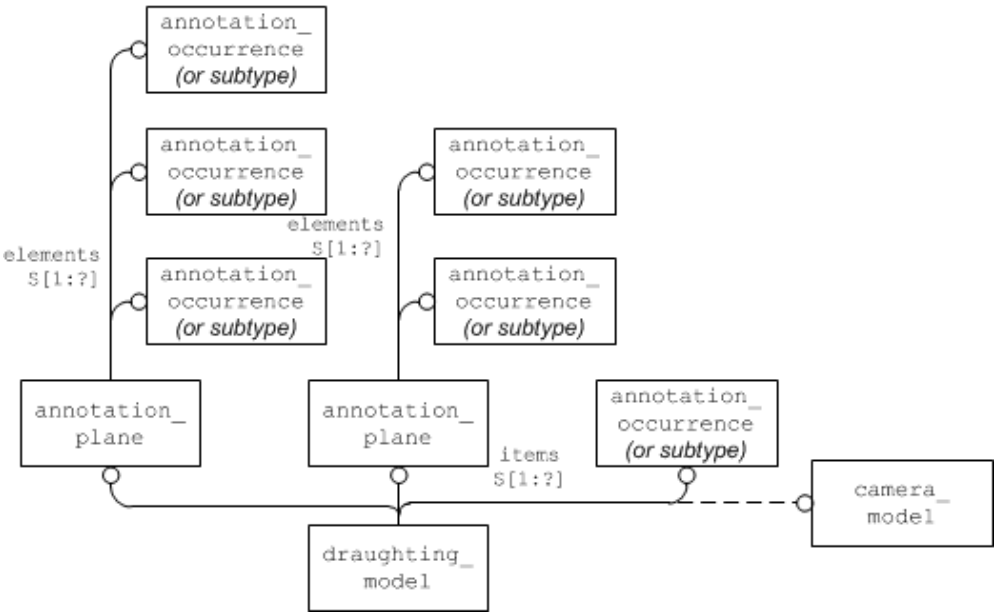


Figure 13: The use of annotation_planes to group some of the annotation_occurrences.

As a result, the convention is that, if an `annotation_plane` is contained in the set of items of a `draughting_model`, all elements contained in the set of elements of the `annotation_plane` are also contained (visible) in the `draughting_model`.

The use of `annotation_plane` to group `annotation_occurrences` is shown in Figure 13 above. This is appropriate for full model definition (see 5.2) or saved view definition (see 5.4.2).

5.4 Saved Views

5.4.1 Definition of Terms

A “saved view” in the context of PMI presentation in STEP as described in this document complies with the following definition taken from ISO 16792, section 5.6:

Saved Views

Saved views of a design model may be defined to facilitate presentation of the model and its annotation. A saved view shall have an identifier, be retrievable on demand, contain a model coordinate system that denotes the direction of the view relative to the model and may contain one or more of the annotation plane(s), a selected set of annotation, or a selected set of geometry.

The following list of equivalent terms might help in technical discussions:

- ISO 16792, ASME Y14.41-2003 - saved view
- CATIA V5 - capture
- NX - work view
- Creo (Pro/Engineer) - combined state

Note: The terms are not 100% equivalent with the Saved View definition. Not all of the capabilities listed above require the definition of a viewing orientation, which is necessary for the full definition of a Saved View. The terms are given here for a better general understanding. Annex B provides more details on how the corresponding CAD system mechanisms are translated to a Saved View in STEP.

5.4.2 Basic Saved View definition in STEP

From the model structure point of view, the definition of a Saved View can be broken down into two major characteristics:

- **what** is being displayed (geometry and annotations)
- **from where** it is being seen (model orientation)

For each of these, there is a key entity in STEP, starting at which the corresponding information can be retrieved:

- the **draughting_model** will define what is being displayed
- the **camera_model** will define from where it is being seen

Note that only the combination of a `draughting_model` with a `camera_model` fulfills all criteria of a Saved View. The `camera_model.name` will carry the name of the Saved View.

A `draughting_model` can also exist as a group of displayed annotations and geometry without an associated `camera_model`, which equals the "capture" concept in CATIA V5. A `draughting_model` may also have several `camera_models` associated to it, which will display the same set of elements from different positions in 3D space.

The base entity to start the definition of a Saved View in STEP is the `draughting_model`. It will define the contents to be displayed.

The `draughting_models` – the global `draughting_model` as well as those defining the scope of visible geometry and annotations for a Saved View – will share the same `geometric_representation_context` as the `shape_representations` defined for the part. See section 5.4.3 below for how these `draughting_models` are related to each other.

It is recommended that the name attribute of the `draughting_model` should be used to convey an identifier for the selected subset of elements. In its set of items, it will reference the geometric and annotation elements to be displayed. To complete the Saved View definition, a `camera_model` will be included in its set of items, which will define the viewing point and the viewing orientation, and carry the name of the Saved View.

The definition of Saved Views in general, however, is not mandatory in most CAD systems. There also may be annotations that are not included in any Saved View, but they still are correctly included in the STEP file structure through the global `draughting_model`.

Definition of the geometry to be displayed

In a Saved View – or a selected subset of the model in general – all or some of the geometric contents will be displayed. There may be more than one geometric representation describing the part. In the current CAx-IF scope, the most likely combination is to have an `advanced_brep_shape_representation` (ABSR) for the solid part shape plus a `constructive_geometry_representation` (CGR) for Supplemental Geometry elements such as reference planes or center axes. There also may be wireframe or surface representations as well, and they all share the same context. Hence, it needs to be identified which of these representations shall be visible in the `draughting_model`, which is done via a `mapped_item` and a `representation_map` (see Figure 14) for each representation of interest. The `axis2_placement_3d`s referenced as a `mapping_source` and a `mapping_origin` shall define a unit transformation.

This allows defining a Saved View which for instance only shows the part shape, not the supplemental geometry, by mapping only the ABSR and not the CGR into the `draughting_model`.

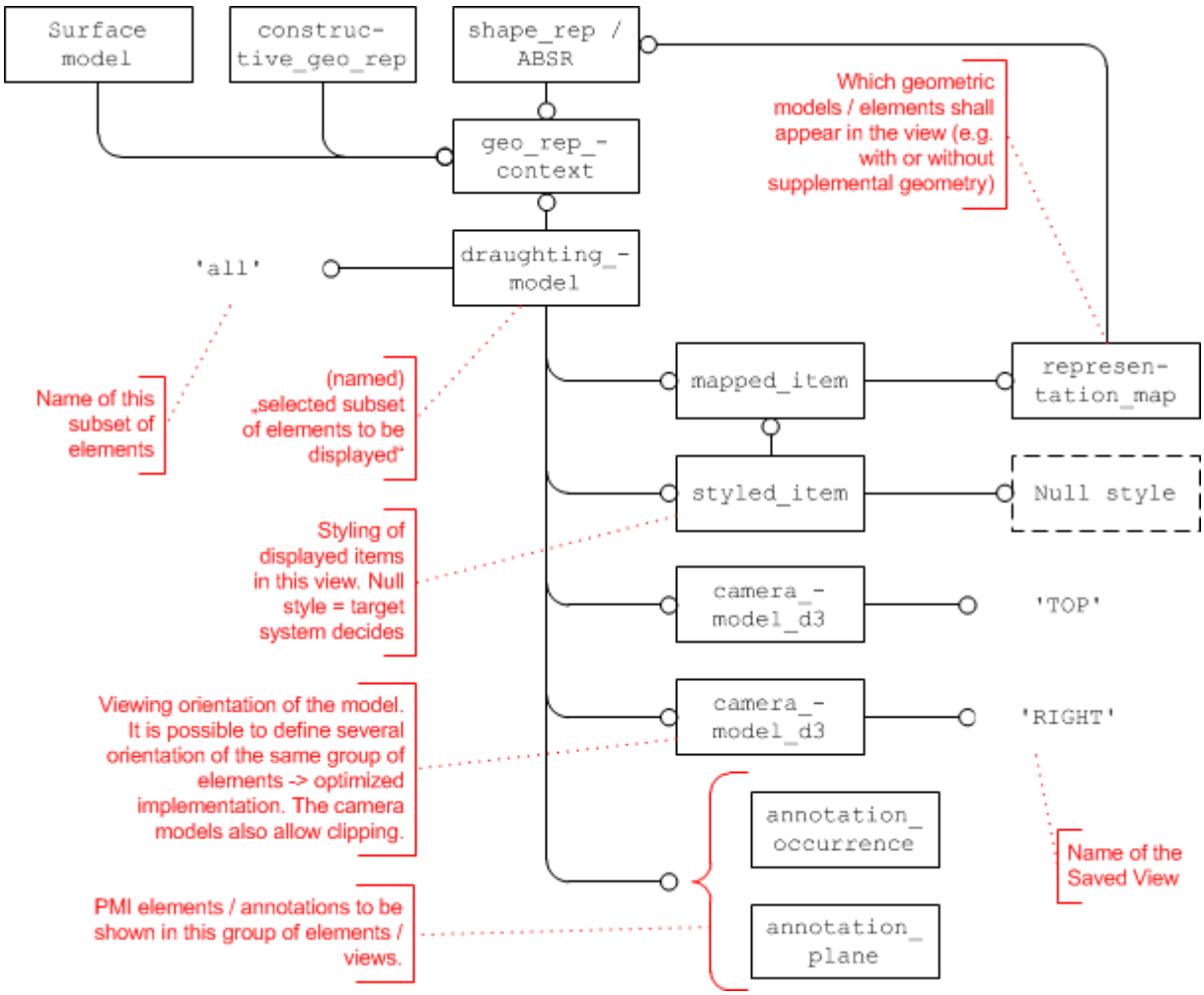


Figure 14: Defining a Saved View with draughting_model and camera_model

As stated in Part 46, only styled items shall be displayed. Hence, in order to convey the information that the target system shall use either the styling defined for the part itself or use its own default styles if none are given in the STEP file, a NULL style shall be assigned to the `mapped_item`.

Note that as stated in 5.1, if the toolkit used in the STEP processor does not allow the creation of a NULL style, a style with an empty color should be created instead. By using this style, it is also possible to style the entire part differently in this specific view.

Also see the “Recommended Practices for Model Styling and Organization” for more information about the usage and interpretation of styles.

Definition of the annotations to be displayed

The second main ingredient in the definition of a Saved View is which annotations – PMI as well as “plain” 3D Text – shall be displayed. This is done by including the desired annotation elements, i.e. the corresponding instances of:

- `annotation_occurrence` (or subtypes)
- `annotation_plane`

into the set of items of the `draughting_model`.

Note that while the first will include individual annotations, including an `annotation_plane` will by definition (see 5.3.3) include all annotations associated with that plane.

Note that some systems offer mechanisms to control the affiliation of items to a defined set using layer assignments and toggling the visibility of these layers rather than managing the individual elements. Though this document will not give a detailed description of this approach (at least for now), the following recommendations are given to ensure interoperability:

- Each layer shall be represented by an instance of `presentation_layer_assignment` with the respective name and the assigned annotations referenced in its set of items.
- The `context_dependent_invisibility` that controls which of the layers will be displayed in the selected set will reference the `draughting_model` as its presentation context.
- The actually displayed set of annotations, which is the result of combining all layer assignments and visibility definitions, shall be included in the set of items of the `draughting_model` in addition, as stated above.

This will allow all importing systems to directly access the information to be displayed, while at the same time preserving the layer definitions for those systems building on them.

Definition of the viewing orientation and zoom factor

As stated in its definition, a Saved View not only refers to a subset of the model geometry and the annotations to be displayed in the view, but also defines the view point and direction of view relative to the model. In the STEP file, this is done by adding a `camera_model_d3` to the set of items of the `draughting_model`, as shown in Figure 15 below.

Note that it is this combination of `camera_model_d3` and `draughting_model` which defines a Saved View in STEP. There may be more than one `camera_model_d3` in the set of items of the `draughting_model`, meaning that the same contents will be displayed from different directions. Hence, the name of the saved view is stored in the camera model's name attribute.

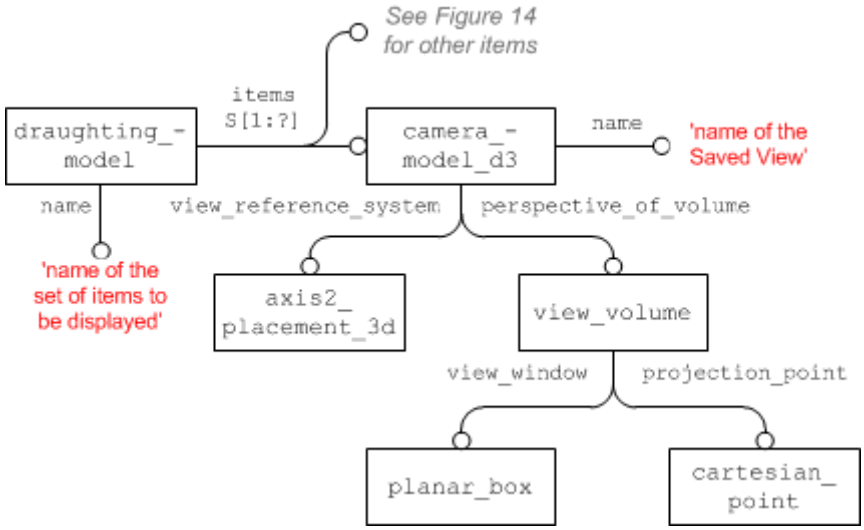


Figure 15: Camera definition for a Saved View

Part21 Example

```
#276=PLANAR_BOX('#276',73.59058615,71.21669627,#275);
#277=CARTESIAN_POINT('#277',(0.0,0.0,0.0));
#278=VIEW_VOLUME(.PARALLEL.,#277,40.620193,40.620193,.T.,162.4
8078,.T.,.T.,#276);
#282=AXIS2_PLACEMENT_3D('#282',#279,$,$);
#283=CAMERA_MODEL_D3('TOP',#282,#278);
#315=DRAUGHTING_MODEL('#315',(#307,#283,#295,#369,#371),#269);
```

Note that in the context of the revision of Part 46, changes have been made concerning the use of `camera_model_d3` and `view_volume`. This Recommended Practice already reflects the agreed changes (see http://www.wikistep.org/bugzilla/show_bug.cgi?id=3293 for details).

Recommendations for practical use:

- the `projection_point` shall be at the origin (0/0/0) of the `view_reference_system`
- the `view_reference_system`'s z axis shall be the viewing direction
- the `view_window` placement shall not be rotated
- Have the location of the `view_window`'s placement so that it centers the `view_window` on (0/0)
- The `view_window` shall be mapped to match the CAD system's window as far as possible while preserving the aspect ratio (see illustration below)
- `view_volume.front_plane_clipping` and `.back_plane_clipping` shall be set to 'false'
- The `view_volume.view_plane_distance` then determines the view distance – hence the zoom factor – of the model display.

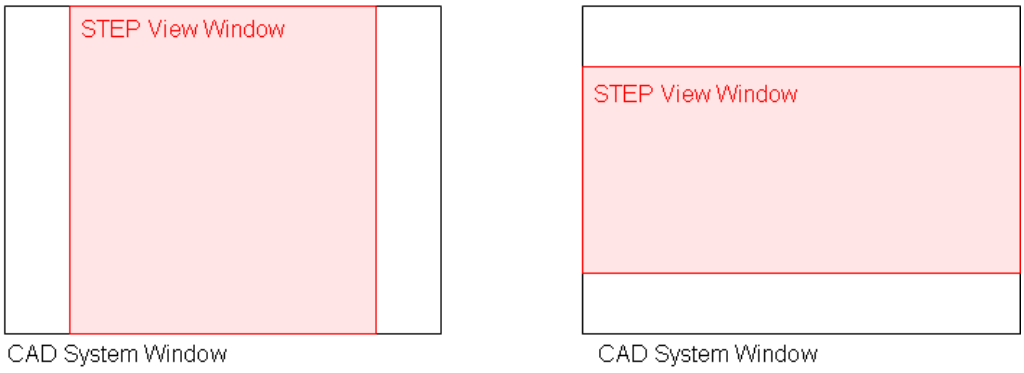


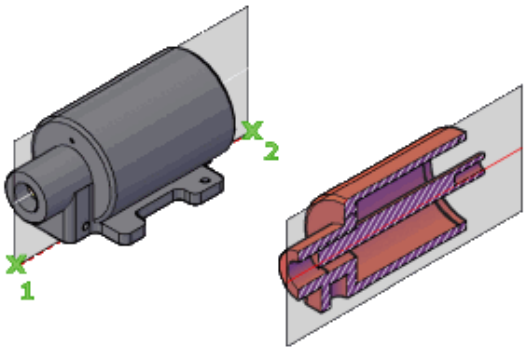
Figure 16: Mapping of the STEP view_window to the CAD system window

The key convention is that the view reference system (VRS) has its origin at the projection point (PP). When this is applied to Figures 11 and 12 in Part 46 Edition 2 (ISO 10303-46:2008) section 4.4.36, it becomes clear that the view plane distance (VPD) equals the distance between eye location and target.

5.4.3 Section Views

For a full 3D description of a part and especially its internal features, it may be necessary to define a Saved View which shows the model cut open in a certain way. PMI data attached to sections in a CAD model are an important way of providing manufacturing information, e.g. on holes.

This capability is usually referred to as “sectioning”. In STEP, it is called “clipping”. In this document, the two terms are used synonymously.



Note: This capability is supported only by AP203e2 (and AP242), since the necessary camera model subtype, `camera_model_d3_multi_clipping`, is not in AP214 (neither IS nor 3rd Edition).

The `camera_model_d3` entity offers even more subtypes which allow for the implementation of further capabilities such as hidden line and surface removal (`camera_model_d3_with_hlhsr`) or light sources (`camera_model_with_light_sources`). These may be considered in the remote future.

Using a single Plane

As stated above, a section view will be defined by using the `camera_model_d3_multi_clipping` entity type. All recommendations concerning viewing orientation and zoom factor given in 5.4.2 still apply. This subtype has an additional attribute to define the section geometry, which in the simplest case is a single `plane`. Figure 17 below shows the entity structure:

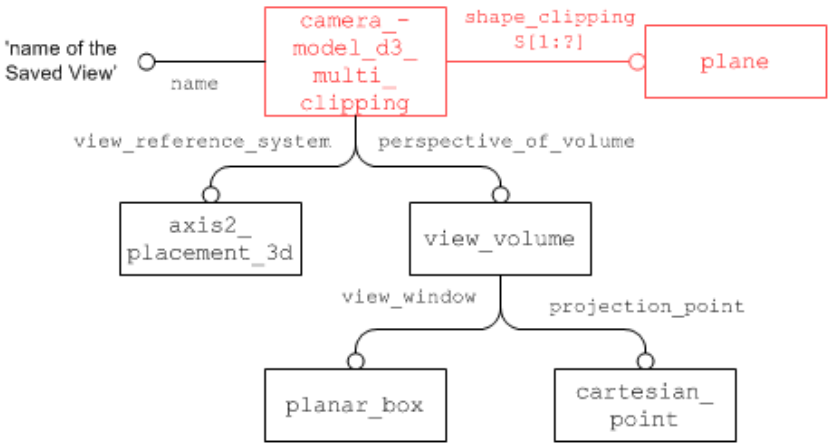


Figure 17: Definition of a Section View using a single Plane

The additional faces that are created in the model where the section geometry cuts the model open will not be transferred as explicit geometry, but shall be determined by the target system. In a later edition of this document, recommendations will be given how these faces can by styled (e.g. with cross-hatching).

Defining advanced Sections

In many cases, more complex geometries are used for clipping. STEP provides a means of defining these by combinations (unions and intersections) of planes. This approach fulfills all requirements given by digital product definition standards (such as ISO 16792). In addition, discussion in the CAx-IF showed that all major CAD systems either use a similar approach, or one that can be unambiguously mapped to and from the STEP approach. In order to ensure interoperability, the following restriction shall be observed when using combinations of planes:

Note: all planes used in the definition of the section geometry for a view shall be perpendicular to one other plane.

This ensures that the resulting section geometry could also be created by extruding a polyline. Some CAD systems even allow arbitrary cutting geometry (e.g. a sphere). This is out of scope.

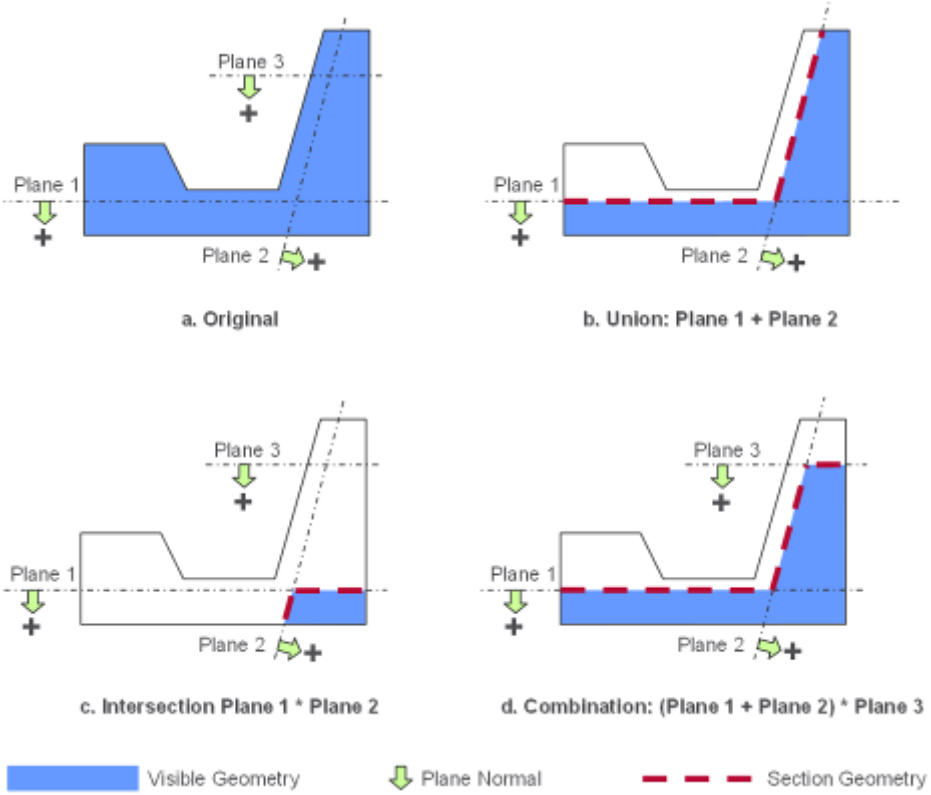


Figure 18: Schematic representation of possible clipping cases

Each plane defines two half spaces (positive and negative), where the positive side is the one the normal is pointing to, as illustrated in Figure 18 above. When the section is applied, the contents of the negative half spaces will be removed, while the contents of the positive half spaces remain visible. There are two operations which allow combining two or more planes:

- Union: any geometry contained in the positive half space of any of the referenced planes is displayed (see Figure 18b)
- Intersection: only geometry contained in the positive half space of all referenced planes combined is displayed (see Figure 18c).

These operations can be combined to build more complex cases. The most famous example for this is the so-called “step”, see Figure 18d. It should be noted that using a mixture of several planes, especially when using intersections, may easily lead to an empty view (no part geometry remains in the resulting “positive” space), so this should be applied carefully.

The STEP entities for these operations are `camera_model_d3_multi_clipping_union` and `camera_model_d3_multi_clipping_intersection` respectively, which can be included in the `shape_clipping` set of `camera_model_d3_multi_clipping`. Figure 19 below illustrates how to implement the “step” example from Figure 18d.

Note the different cardinality of the `shape_clipping` attribute in `camera_model_d3_multi_clipping` and its union / intersection relatives. It is of course also possible to define a union of more than two planes at once, or to use several planes directly in the `camera_model_d3_multi_clipping`.

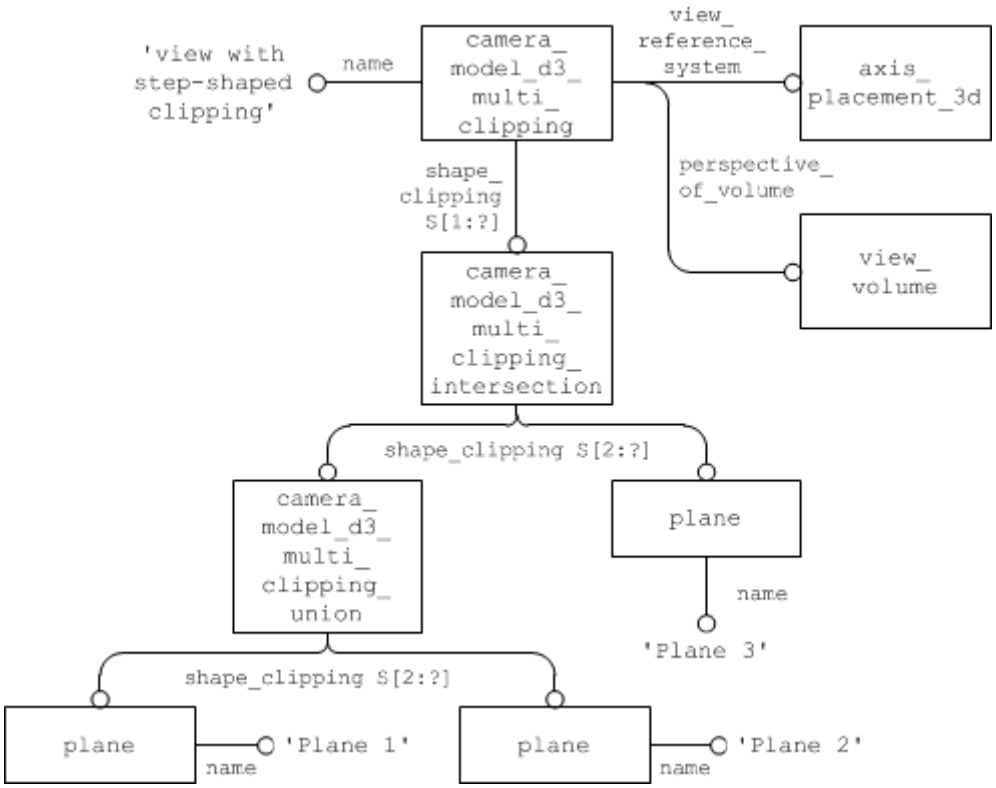


Figure 19: Definition of a Saved View using a Combination of Planes

5.4.4 Relating the draughting_models

As stated before, the named `draughting_model` defines a subset of the geometry and annotations defined in the file. If such a `draughting_model` references an `annotation_plane`, all the `draughting_callouts` in its set of elements are added to the set. Where reference is required for a subset of those annotations linked to an `annotation_plane`, the `draughting_model` references those specific `draughting_callouts` directly.

The global `draughting_model` (see section 5.2) and those defining sets of displayed elements are also put in relation to each other.

Note that annotations may be contained in several Saved Views and hence will be referenced multiple times: once by the global `draughting_model`, and once for each set of items for display they are contained in. In the receiving system, these annotations still shall be instantiated only once.

Each `draughting_model` for a selected set of items has to be related to the global `draughting_model` through a `mechanical_design_and_draughting_relationship` (a subtype of `representation_relationship`), where the global view is always referenced in the `rep_1` attribute, and the saved view in `rep_2` (see Figure 20 below).

Note that the entity type `mechanical_design_and_draughting_relationship` (`MDADR`) is not included in AP214 Edition 3. Hence, the use of the supertype `representation_relationship` is recommended for this purpose in AP214 files, and has to be supported on import into AP203e2 (and AP242).

Note that `MDADR` is a subtype of `definitional_representation_relationship`, for which Part 43 states: “The `representation` indicated by `rep_1` is part of the definition indicated by `rep_2`”. This means that the definition given in this document is actually in the

wrong order. This has been fixed in the Recommended Practices for AP242-based implementations. For the support of legacy files, it was however decided to keep the definition in the reversed order for AP203e2 and AP214e3, keeping that as stated in Table 2 below, the order of `MDADR` attribute is just one of several ways to distinguish between the `draughting_models`.

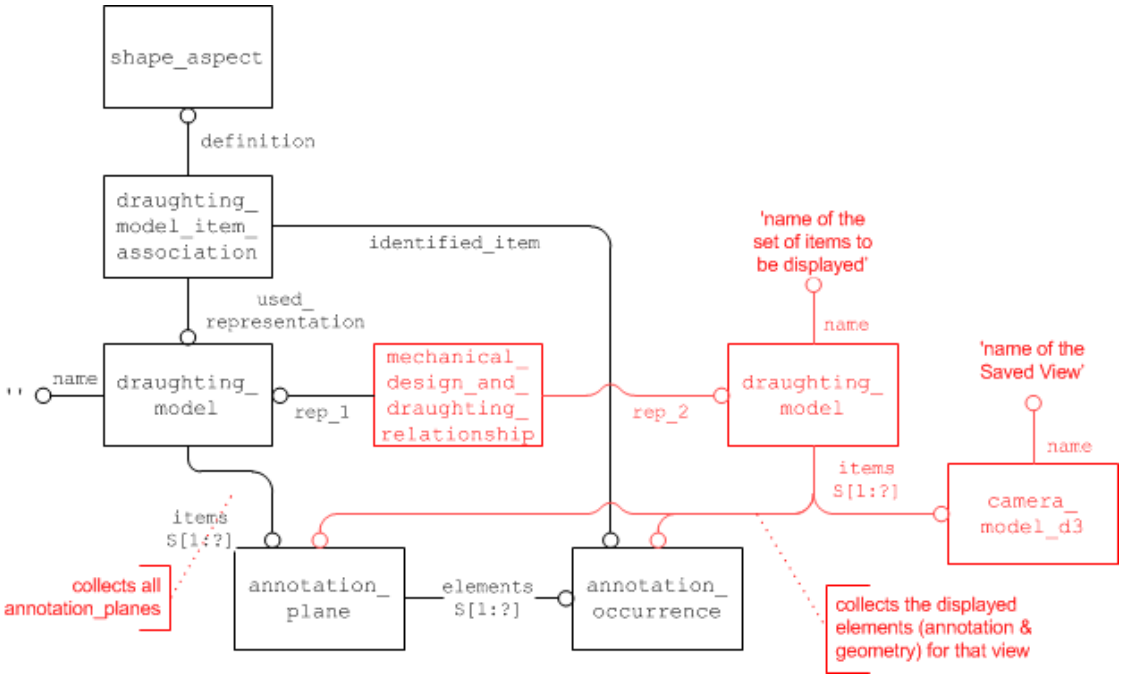


Figure 20: Relating the global and Saved Views

The following table illustrates the differences between the global `draughting_model` and that for a saved view:

Property	Global	Saved View
`draughting_model.name`	'' (empty string)	Name for the set of elements displayed in the view
related annotations	all `annotation_planes`	Some `annotation_planes` and additional annotation elements as per the Saved View definition
`camera_model_d3`	none related	One or several in the set of items
`camera_model_d3.name`		Name of the Saved View as defined in the CAD system
`MDADR` attribute(*)	`rep_1`	`rep_2`
`draughting_model_item_association`	one for each annotation element	None

(*) = These two attributes are reversed for AP242-based implementations to comply with Part 43.

Table 2: Global and Saved View draughting_model properties

5.4.5 Advanced Saved View Implementation

While the basic view definition as given in 5.4.2 provides all the means necessary to define a Saved View as defined by the applicable standards, it has some limitations when compared to CAD system functionalities in this area. The most notable ones are the restriction to display one view at a time and the lack of “flat to screen” notes.

The advanced approach for the implementation of Saved Views builds on the basic approach defined in 5.4.2, but defines the additional data structures necessary to close the gaps mentioned above.

Figure 21 below gives an overview on the complete STEP file structure to define a Saved View in the advanced way. The blue elements in the bottom right hand corner give the basic definition, including the definition of which elements (geometry and annotations) will be displayed. The red elements above that define the additional structure for the advanced implementation.

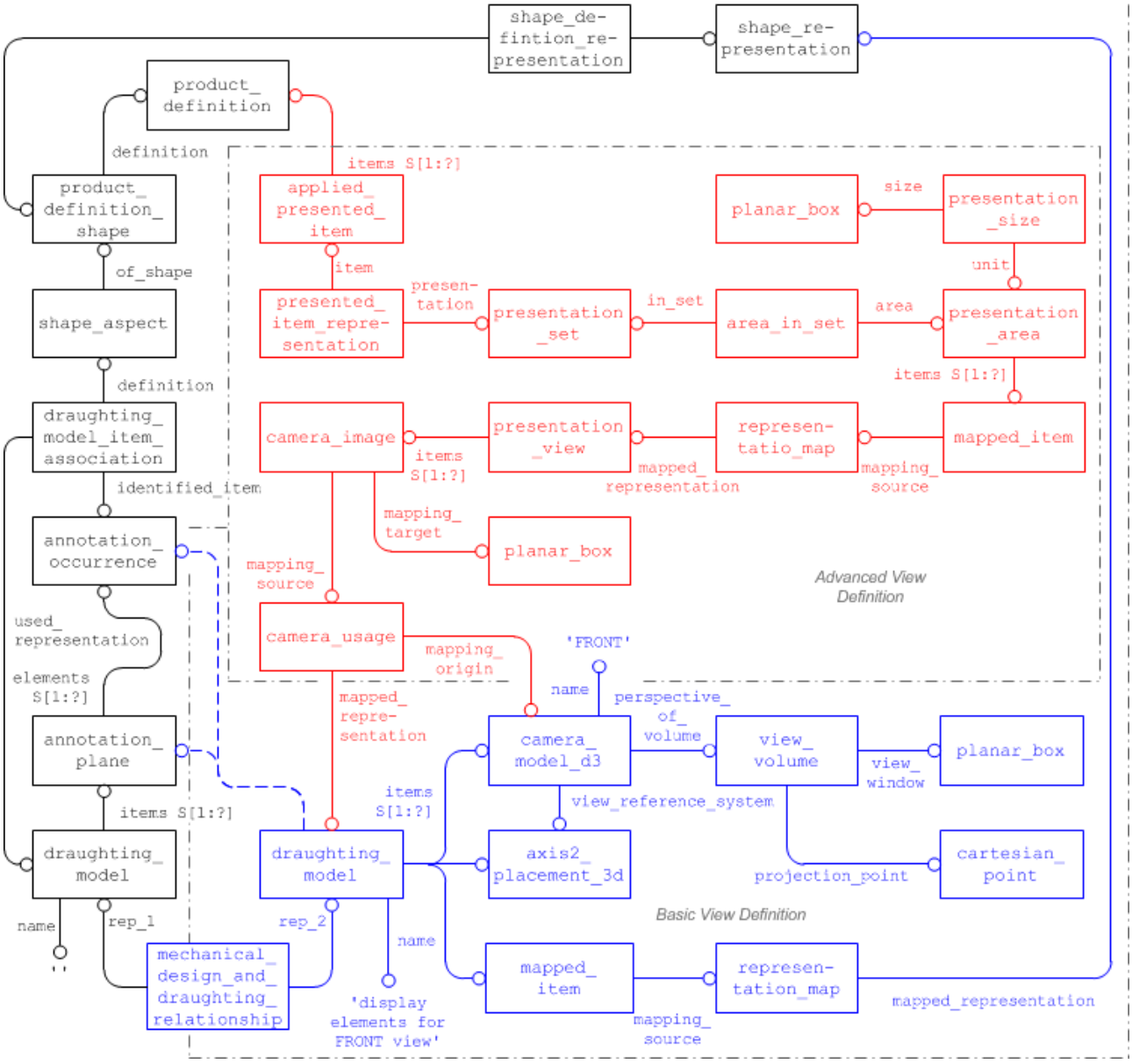


Figure 21: Advanced Saved View Implementation

Recommendations for practical use

- There should be exactly one `presentation_area` in the `presentation_set`. Though it is possible to create manifold views showing several individual views at once, the CAx-IF agreed that for the time being, only one view should be displayed at a time. Hence, the same scaling recommendations as given in 5.4.2 apply to the `camera_image` and the `presentation_size`.
- Instead of the `mapped_item` and `representation_map` used to link the `presentation_area` to the `presentation_view`, a `representation_relationshop` can also be used, where the `presentation_area` is the `rep_1` attribute, and `presentation_view` is `rep_2`.

Flat to Screen Notes

All the annotations and callouts that are items of a `draughting_model` rotate by default together with the 3D model, because they share the same `geometric_representation_context`. But there may be annotations that should always be visible in the 2D foreground plane, in the front of the 3D object. This can be achieved by placing the respective annotations in the set of items of the `presentation_view` in an advanced Saved View.

Flag Notes

Building on this approach, the implementation scope can be extended to support Flag Notes in the future. These consist of two parts, a 3D note (the “flag”) which usually shows a number attached to the 3D part shape, and the description text for this number, which will typically be displayed as a “flat to screen” note. Though the two components can already be implemented independently with the definitions given above, it still needs to be agreed on how the two components can be associated with each other intelligently.

This is out of scope for AP203e2 and AP214e3. It will be supported in AP242 via the “Linking Annotations to other Annotations” capability – see section 5.3.2 above and the comprehensive PMI Rec. Pracs., version 3.6 or later.

6 PMI Presentation Validation Properties

This section concerns the definition of 'validation properties' for PMI data. Similar to the Geometric Validation Properties, they aim to add information about the PMI data contained in the model to the STEP file, so that an importing system has reference points to verify if all data has been processed correctly.

The exporting system will calculate the respective values based on the native model and write them as numerical or string values into the STEP file. The receiving system upon import will re-create the information from the file, and derive the same attributes based on the results. These will be compared with the data given in the file, and if the values match, the import will be deemed successful. The approach used for implementation is based on the "general property" approach defined in Part 41, and widely used of other types of validation properties.

6.1 Validation Properties on the Part/Product level

In order to provide the user with a summary of PMI information in the file, validation properties can be defined at the top-level part (product) in the file.

6.1.1 Total number of annotations per file

The value contains the total number of annotations in the file, regardless of their semantic origin (PMI or other) and which views they are assigned to. The suggested strings to use in the name attributes are shown in Figure 22 below:

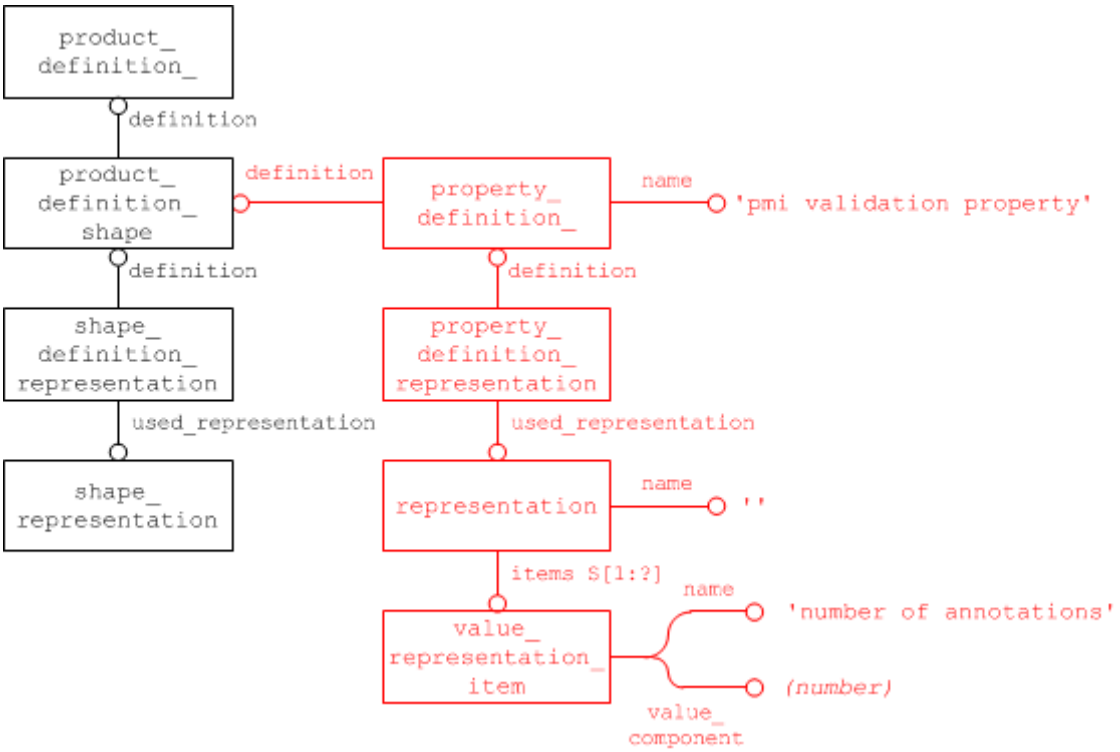


Figure 22: PMI Validation Property for total number of annotations in the file

6.1.2 Total number of views

The total number of views in the file is also linked to the top-level product in the file. The STEP file structure is the same as for the total number of annotations, see Figure 22. The only difference is the suggested strings for the name attributes, which are:

Attribute	Recommended value
property_definition.name	'pmi validation property'
representation.name	''
value_representation_item.name	'number of views'
value_representation_item. value_component	*(number of views)*

Table 3: Attribute values for the 'number of view' validation property

6.2 Validation Properties for Saved Views

The entity that defines which elements are displayed in a Saved View is the draughting_model, see section 5.4.2 for details. Hence, validation properties on a 'per view' basis shall be linked to this entity. In order to define a property on a representation (here, the draughting_model), the following construct is used:

Create a complex instance of draughting_model with characterized_object. Properties can be attached to this complex instance in the usual way. The characterized_object's name and description shall not be instantiated.

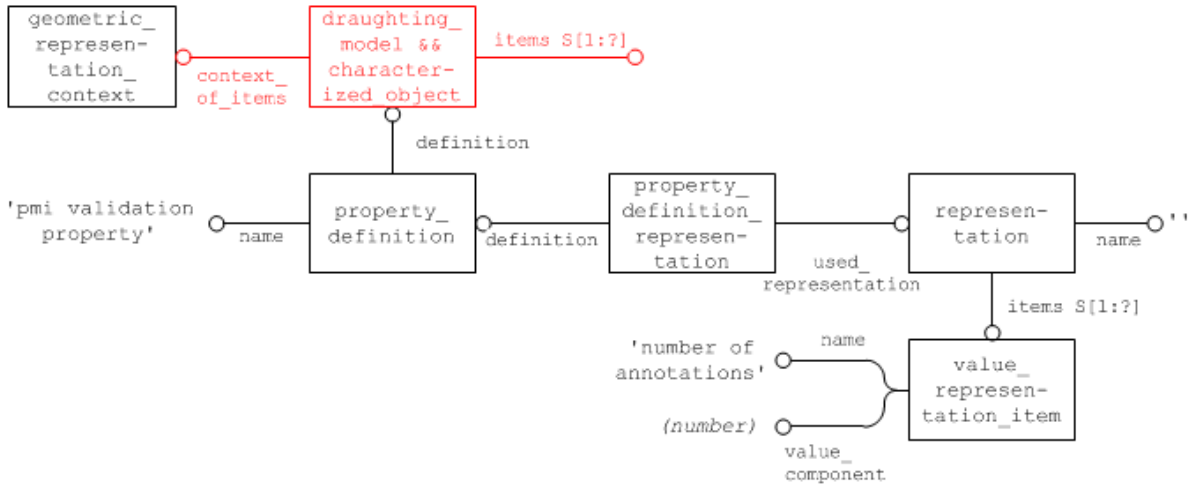


Figure 23: PMI Validation Property for Number of Annotations in a Saved View

Important Note

While this can be implemented as shown above in AP203e2 as well as AP214, this construct does violate a rule in EXPRESS, since characterized_object and draughting_model have no common type in the inheritance tree, and thus are not allowed to be combined into a complex instance. However, it was determined that this is the only way to support this capability in AP203e2 and AP214e3 at all. Based on user requirements, the CAx-IF agreed to hazard the consequences and handle this as a "known issue".

As a solution to this, the new entity type characterized_representation was introduced in AP242, thus legalizing the structure shown in Figure 23. See the comprehensive PMI Rec. Pracs., version 3.6 or later, for implementation details.

Part21 Example

```
#100=(CHARACTERIZED_OBJECT(*,*) DRAUGHTING_MODEL()
REPRESENTATION('PMI_TOP',(#150,#160,#170,#180), #80));
#110=PROPERTY_DEFINITION('pmi validation property','',#100);
#120=PROPERTY_DEFINITION_REPRESENTATION(#110,#130);
#130=REPRESENTATION('',(#140),#80);
#140=VALUE_REPRESENTATION_ITEM('number of
annotations',COUNT_MEASURE(4.0));
```

6.3 Validation Properties for Polyline Annotations

Because of their very nature, validation of Polyline Annotations information content is limited, since once they have been rendered all information about their contents is lost. The validation properties provided for Polyline Presentation are therefore primarily of geometric nature. Content validation information can be supplied by using specific Unicode strings.

The anchor entity for a Polyline Annotation is an `annotation_occurrence` or `annotation_fill_area_occurrence`. In the case when an annotation consists of several subsets, the validation properties shall be consistent with the content of the subset they are attached to.

Important Note

The same limitation that applied to `draughting_models` with regard to attaching properties also applies here, and a similar workaround is proposed for use in AP203e2 and AP214e3: the creation of a complex instance of `annotation_occurrence` or `annotation_fill_area_occurrence` together with `characterized_object`.

This currently violates the same EXPRESS rule as mentioned in section 6.2 above.

This has also been resolved in AP242, but in a different way: the complex entity described here has been replaced by a new intermediate entity, `characterized_item_within_representation`. See the comprehensive PMI Rec. Pracs., version 3.6 or later, for implementation details.

6.3.1 Polyline Curve Length

To make sure that no portions of the rendered annotation are lost, a validation property is defined which contains the total curve length, i.e. the sum of the length of all line segments and circular arcs making up the polyline annotation.

Part21 Example

```
#100=DRAUGHTING_MODEL('',(#150,#160,#170,#180), #80));
#150=ANNOTATION_PLANE('',(#151),#155,(#200,#220,#240));
#200= 200=(ANNOTATION_OCCURRENCE() CHARACTERIZED_OBJECT(*,*)
REPRESENTATION_ITEM('linear
dimension')STYLED_ITEM((#201),#205));
#205=GEOMETRIC_CURVE_SET('linear
dimension',(#206,#207,#208,#209));
#210=PROPERTY_DEFINITION('pmi validation property','',#205);
#211=PROPERTY_DEFINITION_REPRESENTATION(#210,#212);
#212=REPRESENTATION('',(#213),#80);
#213=MEASURE_REPRESENTATION_ITEM('polyline curve length',
POSITIVE_LENGTH_MEASURE(25.4),#50);
```

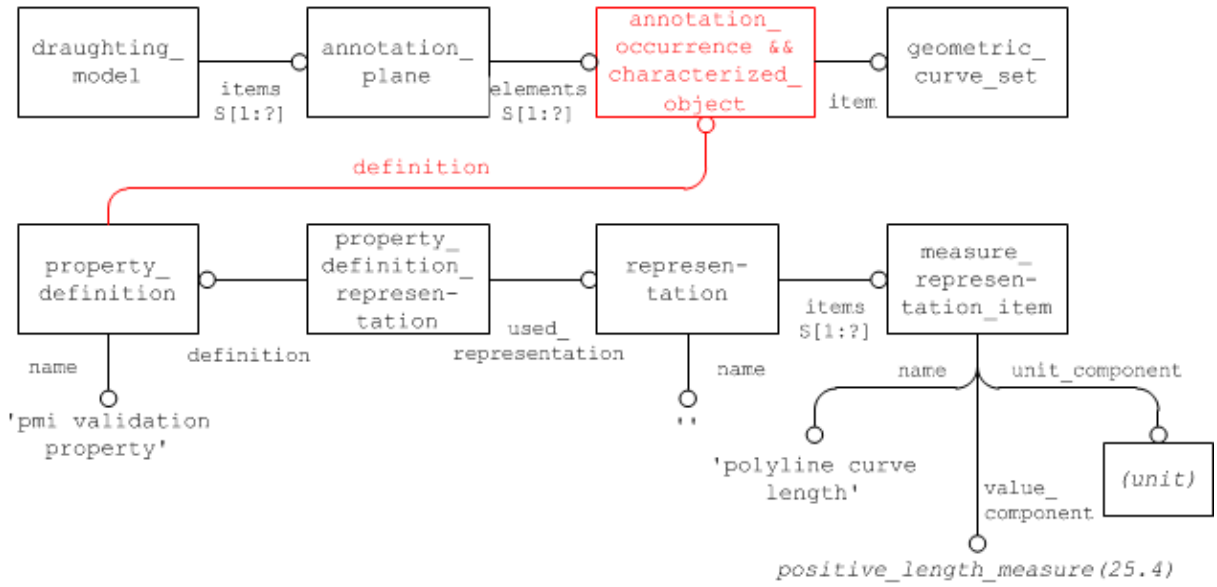


Figure 24: PMI Validation Property for Polyline Curve Length

6.3.2 Polyline Centroid

In order to verify the positioning of the Polyline annotation, a centroid shall be calculated for it, based on the following algorithm:

- Each Polyline annotation consists of line segments and circular arcs.
- The “centroid” of a line segment is the center point of the line.
- The “centroid” of a circular arc is on the radius joining the center of the arc at the position

$$p = \frac{radius * \sin(\frac{arc\_angle}{2}) * 2}{arc\_angle}$$

- For a set of line segments and arcs, the “centroid” is arrived at by a composition of all the element centroids, with the length of each line segment or arc as weight (similar as for a set of solids or surfaces).

Note that since the lengths of the line segments and arcs are used as weights to compose the Polyline centroid, calculating the Polyline Curve Length and the Polyline Centroid validation property at the same time is very efficient.

The approach and file structure used will be similar to the one defined in Section 6.3.1 for the Polyline Curve Length, with the difference being that the item contained in the representation will be a `cartesian_point`, the name of which shall be ‘polyline centre point’:

Attribute	Recommended value
`property_definition.name`	'pmi validation property'
`representation.name`	''
`cartesian_point.name`	'polyline centre point'

Table 4: Attribute values for the 'poyline centroid' PMI Validation Property

6.3.3 Equivalent Unicode String

As PMI data, once rendered into Polylines, is no longer directly machine-readable. A means for validation of the displayed contents is required especially in the context of long-term archiving. Thence, an "equivalent Unicode string" validation property is introduced to support the following validation process:

- A PMI element is converted into a Unicode string. This conversion is independent from the designing CAD system and also the data exchange standard used.

Refer to the "**Unicode String Specification**" (**Revision J**) for the definition of the Unicode strings. It be found on the CAx-IF web sites under "Joint Testing Information":

http://www.cax-if.de/documents/rec_prac_unicode_strings_rev_j_2011-05-25.pdf

- The resulting string will be included as a validation property at the `annotation_occurrence` (subtype) and shall contain the equivalent of what is contained in the respective subset.
- To validate the contents, character recognition software will be applied to the Polyline data, and its output will be compared to the Unicode string(s).
- In order to provide additional information to the character recognition software to achieve better results, it is recommended to also include the font name as defined in the source CAD system for that annotation.

The implementation in STEP is similar to 6.3.1 and 6.3.2; with a `descriptive_representation_item` having a name of 'equivalent unicode string', and the description attribute containing the Unicode string.

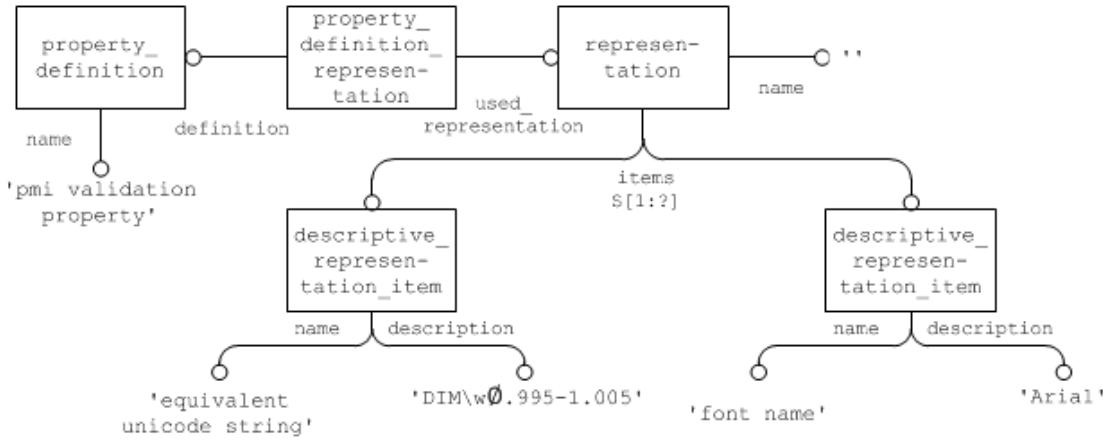


Figure 25: Equivalent Unicode String validation property

6.3.4 Affected Geometry

In addition to validation that the contents are complete, it is also an important part of the PMI definition to which elements in the model a PMI statement applies. Since there are a number of examples where the topologic definition of the geometry changes during the conversion via STEP (the most popular example being a cylinder becoming split into two half-cylinders), it is important to provide a means of validating that the geometry affected by a PMI statement altogether remains unchanged.

Depending on the type of the underlying PMI statement, the "affected geometry" validation property will transfer either:

- the total surface area for PMI associated with faces (e.g. affected by a flatness tolerance), or
- the total curve length for PMI associated with curves (e.g. a dimension).

The STEP file structure to define this validation property is the same as given in 6.3.1 for the Polyline Curve Length, only that the name of the `measure_representation_item` shall be either 'affected area' or 'affected curve length'.

Notes: PMI Annotations can be attached to elements of Supplemental Geometry as well, for instance a reference plane (see corresponding Recommended Practices). The link between an annotation and supplemental geometry works in the same way as between an annotation and part geometry described in section 5.3.1 above.

In some CAD systems, reference planes and axes are defined as unbounded (infinite) elements. In such a case, the “affected geometry” validation property shall not be applied since it would not render meaningful results.

There are two scenarios: the elements are unbounded in the exporting system; then, the validation property shall not be created in the STEP file; second, the exporting system has bounded elements, hence creates the validation property in the file, but the importing system uses unbounded elements and discards the boundaries defined in the file, then it shall not evaluate the validation property.

6.3.5 Combining PMI Validation Properties for Efficient Implementation

Each of the PMI Validation Properties described above follows the exact same pattern for its definition: `property_definiton` ← `property_definition_representation` → `representation` → `representatation_item`. In a larger file, where there are many validation properties – especially if there are many PMI annotations, each with its own set of properties – this can result in a significant number of entities, especially representations, which may have an impact on system performance.

Hence it was agreed that under specific circumstances, multiple validation properties can be combined so they share the same `property_definition`, `property_definition_representation` and `respresentation`. The resulting structure is illustrated in Figure 26 below.

Validation properties can be combined in this way, if they are:

- of the same type (`property_definition.name`)
- attached to the same model element (`property_definition.definition`)

Note that the `representation.name` has to be empty. The individual validation properties can be distinguished by their respective `representation_item.name`. Since each validation property is instantiated at most once per model element, this does not introduce any ambiguities.

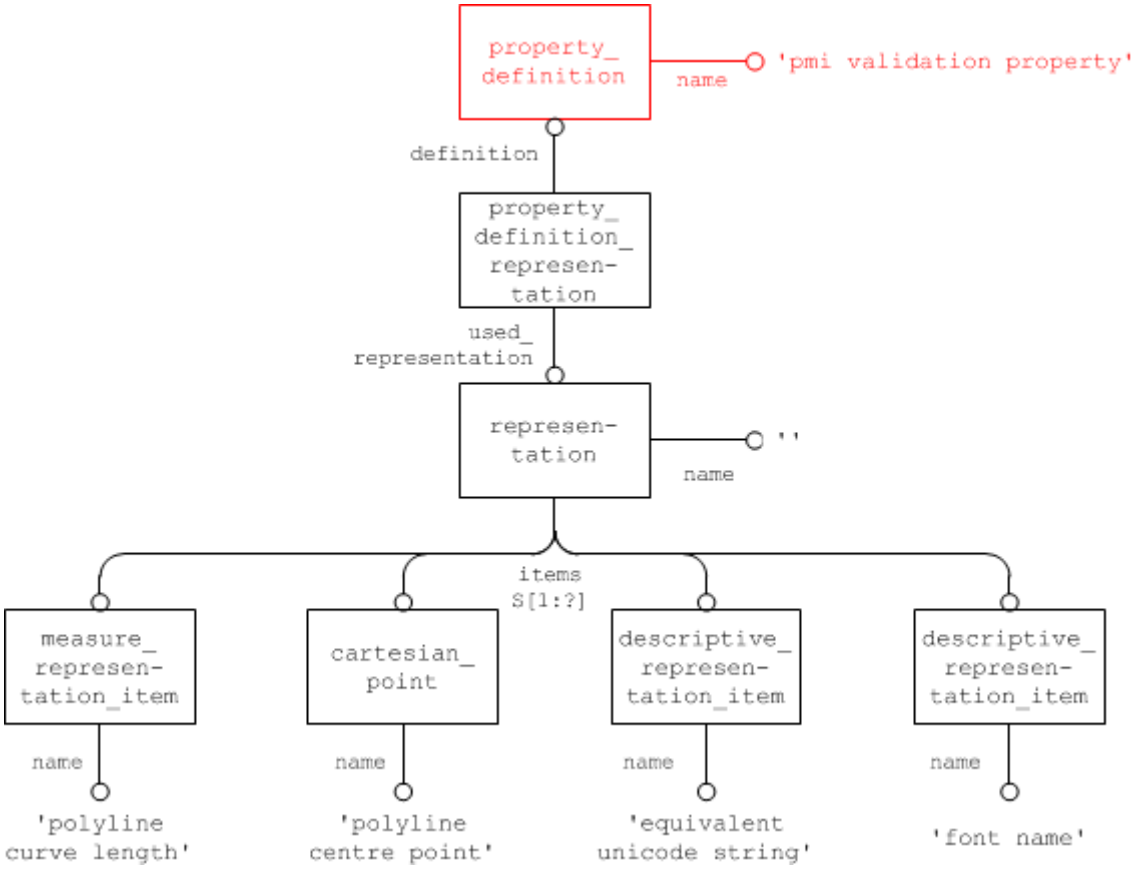


Figure 26: Combining all Polyline Presentation Validation Properties in a single Property

Part21 Example

```
#100=DRAUGHTING_MODEL('',(#150,#160,#170,#180), #80));

#150=ANNOTATION_PLANE('',(#151),#155,(#200,#220,#240));

#200= 200=(ANNOTATION_OCCURRENCE() CHARACTERIZED_OBJECT(*,*)
REPRESENTATION_ITEM('linear
dimension')STYLED_ITEM((#201),#205));

#205=GEOMETRIC_CURVE_SET('linear
dimension',(#206,#207,#208,#209));

#210=PROPERTY_DEFINITION('pmi validation property','',#205);

#211=PROPERTY_DEFINITION_REPRESENTATION(#210,#212);

#212=REPRESENTATION('',(#213,#214,#215,#216),#80);

#213=MEASURE_REPRESENTATION_ITEM('polyline curve length',
POSITIVE_LENGTH_MEASURE(25.4),#50);

#214=CARTESIAN_POINT('polyline centre point',
(0.003890,1.298767,17.390785));

#215=DESCRIPTIVE_REPRESENTATION_ITEM('equivalent unicode
string','DIM\\w\S\X.995-1.005');

#216=DESCRIPTIVE_REPRESENTATION_ITEM('font name','Arial');
```

6.5 PMI Validation Properties Attribute Summary

The following table summarizes the attribute values for all PMI Validation Properties as defined above:

Validation Property	**property_definition.name**	**property_definition.definition**	**representation_item.name and type**
Total Number of Annotations per File	'pmi validation property'	product_definition_shape	'number of annotations' integer_rep_item
Total Number of Views per File	'pmi validation property'	product_definition_shape	'number of views' integer_rep_item
Number of Annotations per Saved View	'pmi validation property'	draughting_model && characterized_object **(*)**	'number of annotations' integer_rep_item
Polyline Curve Length	'pmi validation property'	annotation_occurrence && characterized_object **(*)**	'polyline curve length' positive_length_measure
Polyline Centroid	'pmi validation property'	annotation_occurrence && characterized_object **(*)**	'polyline centre point' cartesian_point
Equivalent Unicode String	'pmi validation property'	annotation_occurrence && characterized_object **(*)**	'equivalent unicode string' descriptive_rep_item
Equivalent Unicode String – OCR Font	'pmi validation property'	annotation_occurrence && characterized_object **(*)**	'font name' descriptive_rep_item
Affected Geometry	'pmi validation property'	annotation_occurrence && characterized_object **(*)**	'affected area' area measure *or* 'affected curve length' positive_length_measure

Table 5: Summary of PMI Validation Property Attributes

Note (*): These entity structures as used here in AP2013e2 and AP214e3 violate a rule in EXPRESS that will be flagged as error by STEP syntax checkers. See highlighted notes in sections 6.2 and 6.3 for details.

Annex A Availability of implementation schemas

AP214 3rd Edition (2010)

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

- http://www.cax-if.de/documents/AP214E3_2010.zip

AP203 2nd Edition (2011)

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 support in the Recommended Practices.

Annex B Definition of Terms for PMI in CAx-IF / LOTAR

The following section provides a harmonized definition of PMI-related terms as they are understood and used within the CAx-IF and LOTAR projects.

B.1 Product and Manufacturing Information (PMI)

Product and Manufacturing Information (PMI) is used in 3D Computer-aided Design (CAD) systems to convey information about the definition of a product's components for manufacturing, inspections and sustainment, which supplements the geometric shape of the product. This includes – but is not limited to – data such as dimensions, tolerances, surface finish, weld symbols, material specifications, 3D annotations and user defined attributes. The term PMI, used by itself, relates to a certain information content within a product definition; i.e. it indicates what information is being stored, independent from how it is being stored.

Though PMI is generally accepted to be the generic designation, the term Geometric Dimensions and Tolerances (GD&T; sometimes also listed as Geometric Dimensioning and Tolerancing) is often used synonymously, as it is the main type of PMI that is currently in focus. Other synonymously used terms are: General Tolerances and Annotations, Annotation, Smart Dimensions, Functional Tolerancing and Annotation (FT&A) or Geometric Product Specification (GPS). Some of these are specific to a particular CAD system. Industry standards for defining PMI include standards such as ASME Y14.5, ASME Y14.41 and ISO 1101, ISO 16792 respectively.

B.1.1 Geometric Dimensions & Tolerances (GD&T)

Geometric Dimensions & Tolerances (GD&T) are a type of Product and Manufacturing Information (PMI) that can be either computed automatically by a CAD system, or entered manually by the user. The definitions below are additions to the terms mentioned in section 3.6 of EN/NAS9300-100:

- *Explicit Tolerance:* Any tolerance with a stated (numeric) value, regardless of how or where it is applied. Explicit tolerances can be applied through general notes, flag notes, PMI or tolerance dimensions. This must be attributable to a specific feature, feature set and/or datum reference (e.g. position, orientation). Standard +/- .03 notes may be explicit, depending on their use.
- *Implicit Tolerance:* Any tolerance where there is no stated value and acceptability of the feature is defined to be through visual comparison to the appearance in the CAD model. Standard +/- .03 notes may also be implicit, depending on their use.
- *Explicit Dimension:* The required nominal value is stated in the CAD model so that it can be obtained without interrogation.
- *Implicit Dimension:* The nominal value can only be obtained by interrogation (i.e. feature to feature measuring) of the CAD model.

B.2 Semantic Representation

Semantic Representation designates a certain way how information is being stored; it does not relate to the information content itself. Semantic Representation captures the meaning (intent) and relationships (context) of a character, word, phrase, sentence, paragraph, specification, or symbol without using any of the visual characters or constructs that are needed for a human to understand it – such as the letters, graphical symbols, lines and arrows used on engineering drawings.

The main purpose of Semantic Representation is to facilitate automated consumption of the data, e.g. for later re-use or for downstream applications. It applies to various types of data, such as PMI, Composite Material Definition, and others.

Example – The Semantic Representation of a Linear Dimension includes all of the information needed to understand the specification (the type of dimension, between which features it is defined…), without any of the graphic components such as dimension lines and extension lines, their direction, arrowheads and the dimension value.

B.3 Presentation

Presentation designates a certain way how information is being stored; it does not relate to the information content itself. Presentation defines the visual representation of a character, word, phrase, sentence, paragraph, specification, or symbol in way that is understandable by humans. Presentation is a generic term that applies to any form of human-readable information transfer; this can for instance be a handwritten note, an engineering drawing, or the display of a 3D CAD model on a computer screen.

The main purpose of Presentation is to facilitate human comprehension of the data, e.g. to manufacture, inspect, assemble or maintain the product described by the data. For a correct interpretation of the presented data, it is required that the reader is familiar with the alphabet used and the general type of information being presented.

In the context of 3D CAD, the term Presentation relates to elements that are visible in the display of a 3D model and are either located (positioned) in 3D space, i.e. they rotate and move with the model, or in a fixed 2D plane. Elements of Presentation can typically by styled (e.g. colored), organized (e.g. in specific views), and associated with other elements of the model. Presented types of data typically are geometry (3D shapes, surfaces, curves, points) and characters (letters, numbers, symbols).

B.3.1 Character-based Presentation

Character-based Presentation is a type of Presentation where the conveyed information is stored as characters (letters, numbers, and symbols). These characters are typically stored in a string variable that can be retrieved and edited in a consuming application. The appearance of Character-based Presentation depends on the font being used and may change if the originating system and the consuming application use different fonts. To ensure no characters are lost from creation to consumption, the alphabet (character encoding) used must be defined as well.

Example – In ASCII, the letter ‘A’ is stored as character code ‘0x41’ (hexadecimal).

Character-based Presentation is often supplemented by geometric elements, such as leader lines, curves or terminator symbols.

B.3.2 Graphic Presentation

Graphic Presentation is a type of Presentation where the conveyed information is converted to geometric elements (lines, arcs, surfaces) by the source system in a way that preserves the exact appearance (color, shape, positioning) of the presented information. The arrangement of these geometric elements can be interpreted by a competent human by looking at them, while the information content is no longer directly computer-accessible.

Example – A simple graphic presentation of the letter 'A' is given by three straight lines. A more complex graphic presentation could have ten straight lines and six circular arcs, but would still be recognizable as an 'A' to a human familiar with the Latin alphabet. In both cases, a computer can only access the geometric definition of the individual elements (start and end coordinates for each line), but not the fact that it is the letter 'A' that is being presented.

Graphic Presentation does not require defining the font or alphabet (character encoding) originally used in the creation of the presented data. In the way Graphic Presentation data is stored, there is typically no distinction between geometric elements that are visual representations of characters, and geometric elements that are visual representations of other constructs, such as leader lines, curves or terminator symbols.

Note – An indirect way of accessing the information content stored as Graphic Presentation is the application of character recognition software that will attempt to identify the original characters from the geometric elements that make up their visual representation. Character recognition, how-ever, has its limitations depending on the algorithms used, the fonts and alphabet involved, and the granularity of the Graphic Presentation geometry elements. Its results cannot be used with the same level of reliability as Character-based Presentation.

B.3.2.1 Polyline Presentation

Polyline Presentation designates a specific implementation form of Graphic Presentation that is supported by many STEP Application Protocols, including AP203e2 (ISO10303-203:2011), AP214e3 (ISO10303-214:2010) and AP242 (ISO10303-242:2014). It supports all the characteristics of Graphic Presentation. A Polyline is defined as an ordered list of 3D points, which are consecutively connected by straight line segments. Circles and circular are the only other allowed geometric elements, and can be used in combination with Polylines. Filled areas can be defined with the aforementioned elements as boundaries.

B.3.2.2 Tessellated Presentation

Tessellated Presentation designates a specific implementation form of Graphic Presentation that is has been introduced during the development of STEP AP242 (ISO10303-242:2014). It supports all the characteristics of Graphic Presentation. It is based on data model for tessellated geometry and provides more efficient ways of storing the data, compared to Polyline Presentation. It supports curves (composed of straight line segments) and surfaces (composed of triangles).

Annex D Mapping of Saved Views

The following section provides an overview at a user-interface level on how the respective native CAD constructs can be mapped to and from a STEP Saved View (see section 5.4.2), considering the standardized definition.

The descriptions currently include Dassault Systèmes' CATIA V5/V6, Siemens' NX and PTC's Creo (previously Pro/Engineer), and can be extended to cover further systems in the future if needed.

Data Format	**STEP AP 203e2, 214e3 & 242**	**CATIA V5/V6 (Dassault Systèmes)**	**NX (Siemens)**	**Pro/Engineer, Creo (PTC)**
View Designation	Draughting Model with included Camera Model	View Annotation Plane Capture	Model View Camera	Combined States Views + Layers
Associated Data		Name of the camera	Drawing Reference x,y,z Vector	
	View Reference System	Point of view	Perspective Distance	Position (horizontal / vertical)
	Camera Model View Window View Volume	3D Shape orientation	Camera: - Camera Name - Camera Point - Target Point - Target Matrix - Magnification	Spin x,y,z
	View Plane Distance	Zoom factor	Display Scale	Zoom factor
	Draughting Model	Active View state		
	Camera Model Name	View Name String	View Name String	View Name String
Data Format	**STEP AP 203e2, 214e3 & 242**	**CATIA V5/V6 (Dassault Systèmes)**	**NX (Siemens)**	**Pro/Engineer, Creo (PTC)**
Differences	Views are created by a combination of Draughting Model and Camera Model. The DM controls visibility of elements (PMI and geometry), the CM controls model orientation and zoom factor.	Views can be stored as "Annotation Views" or as "Capture Views". Specific PMI can be associated with both types. Captures can be associated with cameras, Annotation Views cannot.	There are Model Views and Cameras, which are both automatically created. In all camera positions all PMI are visible. Other cameras can be created by the user.	Only Views can be created. In contrast to other systems, the Views are not shown in the model tree. The PMI are not directly associated with the views; the association is done using Layers. Combined States are used to combine Views and Layers, hence PMI.

Table 6: General Overview on Views in JT and major CAD systems

Remarks concerning CATA V5/V6

The main point to consider when translating data from CATIA V5/V6 to STEP is that in CATIA, there are two source constructs which can be mapped to Saved Views: Captures and Views/Annotation Planes.

When creating PMI (or annotations in general) in CATIA, each PMI belongs to exactly one View (the one active during the PMI creation). Views/Annotation Planes are specified around the geometry for automated generation of views, sections and cuts (with a 2D drawing in mind), i.e. each annotation is either on the associated annotation plane, or parallel to it.

Each PMI may belong to 0,1 or N Captures. Displaying a tolerance capture means showing annotations and/or annotation planes (hence all annotations associated with that annotation plane) according to a specific context. The user can record what is shown/not shown in the capture, and can also define a point of view by associating a camera.

Views/Annotation Planes and Captures both are capable of defining section cut views by associating clipping planes.

Captures can be mapped to a fully defined Saved View (and vice versa), which also satisfies the definition per ISO 16792. This is the recommended approach.

Views/Annotation Planes can also be mapped to Saved Views, but provide only a partial specification since a camera definition is missing. This means that the `draughting_model` could be fully populated, but would miss a `camera_model_d3`. The definition of Annotation Plane renders a “view axis” (the normal vector of the plane) which could be used to “guess” a view point on that view axis, but as a result the zoom factor in STEP will almost always be different than in the native CATIA model, or the model is even seen as from behind.

On import into CATIA, the general approach is to automatically create an annotation plane for each annotation; i.e. in a CATIA – STEP – CATIA round-trip, the number and setup of Views/Annotation Planes will be different in the resulting model compared to the original model, but the Captures (visible elements, orientation, naming) should be preserved.

Remarks concerning NX

A set of standard Model Views (“Back”, “Front”, “Bottom”, “Isometric”, “Trimetric”, “Left”, “Right”, “Top”) is automatically created in NX when creating a part. These names cannot be changed, but additional model views can be created interactively, with user-defined names.

The camera is defined by the zooming and rotating the model to the desired position and saving the model view. NX will automatically add the camera to the model view. The camera handle can be selected and moved in space to setup an eye direction.

When PMIs are created, they are by default associated with the Model View that is currently set as active, and will be visible in this view only.

Remarks concerning Creo / Pro/Engineer

In Creo and Pro/Engineer, two data structures need to be combined to generate a Saved View: Views and Layers:

- A View controls how the model is displayed, i.e. model orientation and zoom factor.
- Layers control which elements of the model (geometry, annotations…) are visible.

The View Manager in Creo / Pro/Engineer allows defining Combined States, which are composed of two or more display state elements such as Model orientation, Model style, Cross section, Layer state, and others.

This means that geometry and PMI (annotations in general) will be assigned to layers, and the layer state defines which of these are visible. This will then be stored in a Combined State, that also includes a view for orientation and zoom facture, and can be given a user-defined name. The mapping of the layer approach to the `draughting_model` used in STEP is defined in section 5.4.2 under “Definition of the geometry to be displayed”.