Recommended Practices

for

3D Tessellated Geometry

Release 1.0

December 17, 2015

Contacts	
Jochen Boy ProSTEP iViP Association Dolivostraße 11 64293 Darmstadt / Germany jochen.boy@prostep.com	Phil Rosché ACCR / PDES, Inc. 125 King Charles Circle Summerville, SC 29485 USA phil.rosche@accr-llc.com
Alain Roche Dassault Systèmes alain.roche@3ds.com	

Table of Contents

1 Introduction 4

2 Scope 5

4 Fundamental Concepts 5

5 3D Tessellated Geometry Description 6

5.1 Coordinates list 6
5.2 3D Wireframe tessellated geometry 6
5.3 3D surface tessellated geometry 6
5.4 3D Tessellated Structured Geometry 9

6 Watertight Tessellation 14

7 3D Tessellated Geometry resulting from an assembly 14

8 Tessellated Validation Properties 16

8.1 Fundamental Concepts 16
8.2 Definitions of Units 16
8.3 Tessellated Validation Properties at the Part level 16
8.4 Validation Properties at the Tessellated Geometry Level 18
8.5 Combining Validation Properties for Efficient Implementation 28
8.6 Evaluation of the Validation Properties 29
8.7 Assembly Validation Properties 29
8.8 Summary of Imposed Attributes Values 30

Annex A Availability of implementation schemas 31

List of Figures

Figure 1: Housing model as 3D Tessellated Geometry 4
Figure 2: Cube as Tessellated Surface Set 8
Figure 3: Triangle strips and triangle fans 8
Figure 4: Tessellation with a triangle strip and an independent triangle 8
Figure 5: Example of a basic solid having 2 planar faces and 2 cylindrical faces 9
Figure 6: Structured 3D Tessellated Geometry with exact geometry information 10
Figure 7: S1 model without normals 10
Figure 8: One coordinates_list per tessellated face 12
Figure 9: One single coordinates_list for a whole solid 12
Figure 10: Structured 3D Tessellated Geometry with BREP exact geometry 13
Figure 11: Watertight tessellation by using connecting edges 14
Figure 12: 3D Tessellated Geometry resulting from an assembly 15
Figure 13: Tessellated Validation Properties at the Part Level 17
Figure 14: Tessellated Validation Property "Bounding Box" 18
Figure 15: Validation Properties at the Tessellated Geometry level 19
Figure 16: Examples of tessellated geometry for Number of Facets 20

Figure 17: Tessellated Validation Property "Number of Facets" 21
Figure 18: Calculation of Area and Centroid for a Facet 21
Figure 19: Tessellated Validation Property "Tessellated Surface Area" 22
Figure 20: Tessellated Validation Property "Tessellated Surface Centre Point" 23
Figure 21: Tessellated Validation Property "Number of Segments" 24
Figure 22: Tessellated Validation Property "Tessellated Curve Length" 25
Figure 23: Tessellated Validation Property "Tessellated Curve Centre Point" 26
Figure 24: Tessellated validation property “tessellated point set centre point“ 27
Figure 25: Combining number of facets, tessellated surface area and tessellated surface centre point 28
Figure 26: Table of Imposed Attributes Values 30

Document History

Revision	Date	Change
0.1(*)	2013-06-06	Initial creation
0.2(*)	2013-09-23	Corrections
0.3(*)	2014-02-27	Rework of the document - Merge with Validation Properties document
0.4(*)	2014-09-14	Update of Validation Properties - Workaround for placement
0.5(*)	2015-02-19	Update of figures for Validation Properties - Editorial changes
0.6(*)	2015-12-03	Update of the chapter about watertight tessellation – Update of Figures
1.0	2015-12-17	Public Release of Version 1.0

(*): Internal review versions; not published.

1 Introduction

Industry requires the capability to exchange tessellated boundary representation (BREP) data via STEP, to enable design-in-context in scenarios. The approach that is currently being developed includes tessellation for 3D solids, independent surfaces, independent wireframes and independent points. In the first stage, all tessellations will be based on triangles. In the second stage, this will be extended to support tessellation based on polygons.

The real-time rendering of 3D models is usually done by using 3D Tessellated Geometry which is made up of data sets of triangles that can be interpreted by graphic libraries such as OpenGL or DirectX.

STEP is widely used for the exchange and long term archiving of CAD models. The need to manage 3D Tessellated Geometry in STEP appeared for the following applications:

- Electrical Harness designed by using 3D Tessellated Geometry as context data: The Long Term Archiving of the Electrical Harness as geometry requires taking into account not only the 3D Exact Geometry but also the 3D Tessellated Geometry.

- Explicit Representation of Composites data: the Composites data model is implicit, i.e. a sequence of plies is mainly represented by a surface, an ordered set of contours lying on the surface and a thickness. There is the requirement for the exact representation of plies as 3D Tessellated Geometry and to exchange this explicit representation in STEP.

- Pre-visualization of STEP data: The conversion of STEP geometry may take a long time. The introduction of a 3D Tessellated Geometry as an alternative representation in addition to the 3D exact geometry provides great benefits:

    - Ability to have a quick pre-visualization of the STEP data before the conversion of the exact data.
    - Ability to check whether the conversion of the exact data is correct or not. This can be done manually by the user or automatically by the translator.

- Quick review of PMI while preserving the cross-highlighting with associated geometry without having to exchange exact geometry (IP protection).
-

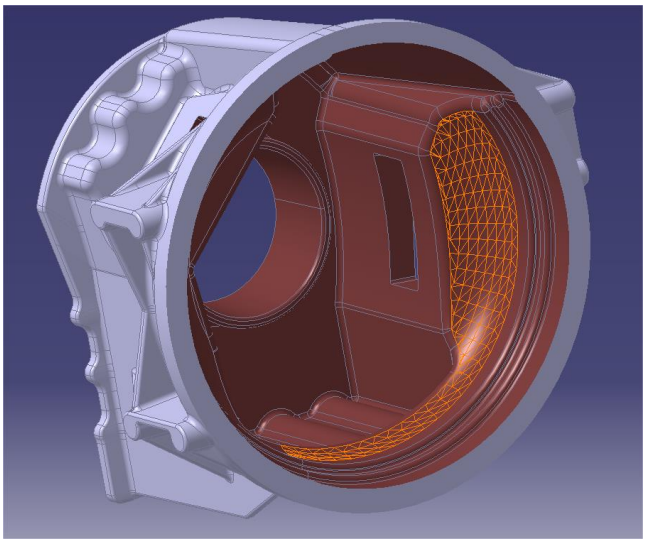

Figure 1: Housing model as 3D Tessellated Geometry

2 Scope

The following is within the scope of this document:

- The definition of 3D Tessellated Geometry in STEP.
- The definition of validation properties for 3D tessellated data in STEP, based on the approach first applied in STEP AP242.
- The assignment of these validation properties to the model (part / product level).
- The assignment of these validation properties to the geometry (tessellated elements).

The following is out of scope for this document:

- The definition of Tessellated PMI Presentation.
- The definition of 2D Tessellated Geometry.
- The way to manage alternative shape representation (exact and tessellated, levels of details).
- The definition of textures, colors, normal on points.
- The definition of scan results.
- The definition of scenes.

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.---3D Tessellated Geometry---1.0---2015-12-17`

It will appear in a STEP file as follows:

```
FILE_DESCRIPTION(('...','CAx-IF Rec.Pracs.---3D Tessellated Geometry---1.0-
--2015-12-17',),'2;1');
```

4 Fundamental Concepts

The 3D tessellated geometry uses sets of points to describe surfaces by using triangles (sets of 3 points), or curves by using polylines (list of n points).

3D tessellated geometry can be represented in AP203 and AP214 as faceted geometry with one `face`, one `polyloop`, and three `cartesian_points` per triangle. However, the size of the STEP files becomes very large so that models cannot be efficiently managed.

3D Tessellated Geometry is based on new entities in ISO 10303 Part 42 geometric resources that can be used in AP242 Edition 1.

The subtype of representation to collect 3D Tessellated Geometry is `tessellated_shape_representation`. A `tessellated_shape_representation` may contain 3D tessellated geometrical entities such as solids, shells, or points.

Note: In AP242 Edition 1, it is not possible to include an `axis2_placement_3d` in a `tessellated_shape_representation`. This issue is documented by the following Bugzilla entry: http://www.wikistep.org/bugzilla/show_bug.cgi?id=5009

Until the issue is resolved, the recommended workaround is to use a plain `shape_representation` instead of a `tessellated_shape_representation` (for example, when it is involved in the definition of an assembly)

5 3D Tessellated Geometry Description

5.1 Coordinates list

The `coordinates_list` entity is the basic entity. It defines a set of 3D points thru its attribute `position_coords`. Each tessellated item refers to its points thru a `coordinates_list` entity. A `coordinates_list` entity can be shared between several tessellated items.

```
ENTITY coordinates_list
    SUBTYPE OF (tessellated_item);
    npoints: INTEGER;
    position_coords: LIST [1:?] OF LIST [3:3] OF REAL;
WHERE
    WR1: npoints = SIZEOF(position_coords);
    WR2: SIZEOF (['GEOMET-
RIC_MODEL_SCHEMA.REPOSITIONED_TESSELLATED_ITEM'] * TYPEOF(SELF)) =
0;
 END_ENTITY;
```

5.2 3D Wireframe tessellated geometry

A set of points is represented by a `tessellated_point_set` entity.

A set of polylines can be represented by a `tessellated_curve_set` entity.

5.3 3D surface tessellated geometry

Tessellated surface geometry can be represented by a `tessellated_surface_set` that can be instantiated by either a `triangulated_surface_set` or by a `complex_triangulated_surface_set`.

```
ENTITY tessellated_surface_set
     ABSTRACT SUPERTYPE OF (ONEOF (triangulated_surface_set, com-
plex_triangulated_surface_set))
     SUBTYPE OF (tessellated_item);
     coordinates: coordinates_list;
     pnmax: INTEGER;
     normals: LIST [0:?] OF LIST [3:3] OF REAL;
WHERE
     WR1: ((SIZEOF(normals) = 0) OR (SIZEOF(normals) = 1) OR
(SIZEOF(normals) = pnmax));
END_ENTITY;
```

```
ENTITY triangulated_surface_set
      SUBTYPE OF (tessellated_surface_set);
      pnindex: LIST [0:?] OF INTEGER;
      triangles: LIST [1:?] OF LIST [3:3] OF INTEGER;
WHERE
      WR1: ((SIZEOF(pnindex) = 0) OR (SIZEOF(pnindex) =
SELF\tessellated_surface_set.pnmax));
      WR2: NOT((SIZEOF(pnindex) = 0) AND
(SELF\tessellated_surface_set.pnmax <> coordinates.npoints));
END_ENTITY;


ENTITY complex_triangulated_surface_set
      SUBTYPE OF (tessellated_surface_set);
      pnindex: LIST [0:?] OF INTEGER;
      triangle_strips: LIST [0:?] OF LIST [3:?] OF INTEGER;
      triangle_fans: LIST [0:?] OF LIST [3:?] OF INTEGER;
WHERE
      WR1: ((SIZEOF(pnindex) = 0) OR (SIZEOF(pnindex) =
SELF\tessellated_surface_set.pnmax));
      WR2: NOT((SIZEOF(pnindex) = 0) AND
(SELF\tessellated_surface_set.pnmax <> coordinates.npoints));
      WR3: NOT ((SIZEOF(triangle_strips) = 0) AND
(SIZEOF(triangle_fans) = 0));
END_ENTITY;
```

5.3.1 Attributes of a tessellated surface set

- **pnmax** is the number of couples (point, normal) involved in the definition of the `tessellated_surface_set`. It is possible to define several normals for a point. This is useful when the underlying surface is not C1 continuous.

- **normals** is the list of normals of the `tessellated_surface_set`. The size of the list of normals may be:
  - **0**: no normals are defined.
  - **1**: the tessellated surface set is planar with only one normal.
  - **pnmax**: each point has a normal.

- **pnindex** is the table of indices of the points used in the definition of the triangles. It is an index to the `coordinates_list`. Its size may be:
  - **pnmax:** this is the size of normals when each point has a normal.
  - **0**: no indirection.

- **triangles** is the list of triangles; each triangle is defined by 3 indices in the table `pnindex`. For each index we get the point from the `coordinates_list` and the normal from the list of normals (if there are `pnmax` normals in the list).

A `tessellated_surface_set` represents any kind of tessellated surface.

The definition of the normals is optional. If no normals are defined, `pnmax` is the number of points used in the surface set.

Examples:

A cube can be represented by a `coordinates_list` entity with 8 points, a `triangulated_surface_set` with `pnmax` = 24, 24 normals, and a list of 12 triangles.

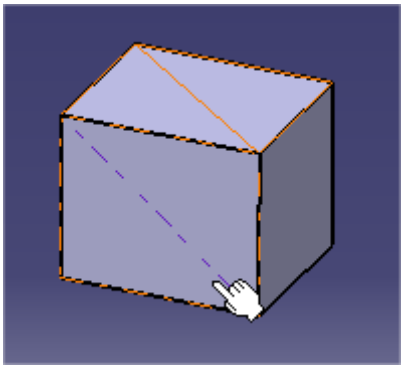

Figure 2: *Cube as Tessellated Surface Set*

A tessellated surface set can also be a set of triangles without any constraint of connectivity between the triangles and without normals. A `tessellated_surface_set` is a collection of triangles and can represent PMI text which is not a connected surface.

The graphical attributes can be defined for the entire tessellated surface set, but not for individual triangles or a set of triangles. For example, in the cube above, you cannot set a color to a side of the cube.

5.3.2 Complex triangulated surface set

In some cases, the tessellation is optimized by using specific sets of triangles named triangle strips or triangle fans.

A triangle strip is a list of 3 or more points defining a set of connected triangles (see Figure 3).

A triangle fan is a set of 3 or more points defining a set of connected triangles sharing a common vertex (See Figure 3).

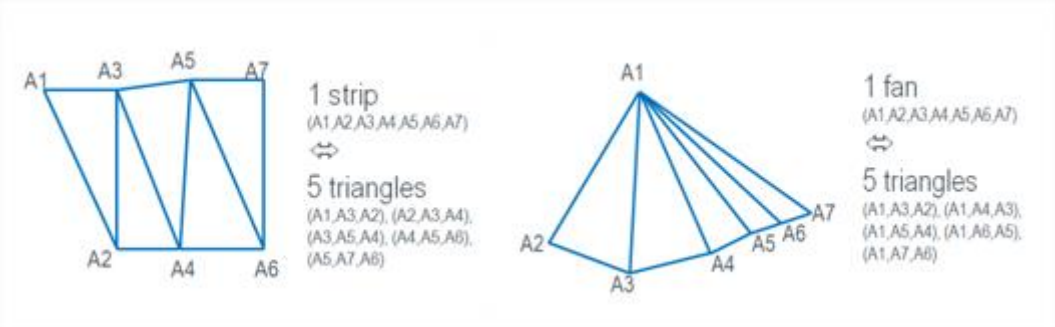

Figure 3: Triangle strips and triangle fans

Note: There is no attribute for independent triangles in a complex triangulated surface set. It is not used because an independent triangle is just a triangle strip with 3 points.

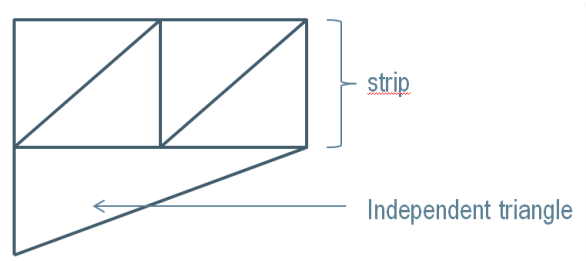

Figure 4: Tessellation with a triangle strip and an independent triangle

5.4 3D Tessellated Structured Geometry

A common use case is that tessellated geometry is derived from exact geometry.

The tessellated structured geometry is intended to preserve the topological structure of the exact geometry. This is a key capability for several use cases:

- Graphical attributes of the exact geometry can be used in the same way.
- PMI Presentation may be linked to the tessellated geometry.

While it is possible to export exact geometry as a tessellated surface set, it is recommended to export it as tessellated structured geometry.

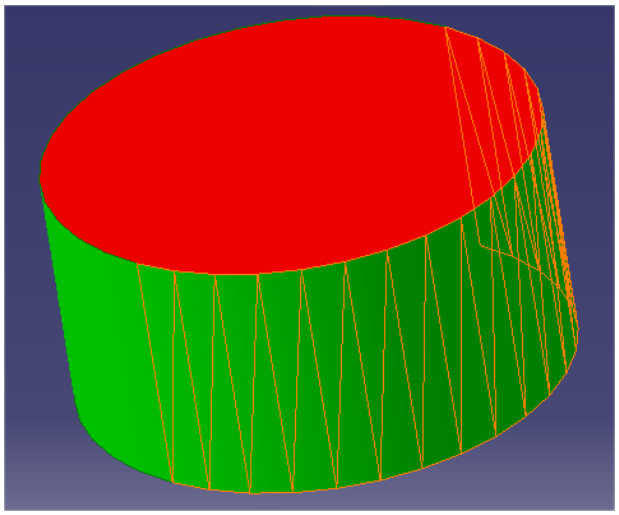

Figure 5: Example of a basic solid having 2 planar faces and 2 cylindrical faces

The 3D tessellated structured geometry reflects the structure of the exact geometry:

- There is one tessellated entity per exact geometric entity (solid, shell, wireframe or point).
- A tessellated solid or a tessellated shell may contain tessellated faces, tessellated edges, and tessellated vertices.
- A tessellated wireframe may contain tessellated edges and tessellated vertices.

A tessellated face can be defined either by a `triangulated_face` or by a `complex_triangulated_face`. These entities are similar to the entities `triangulated_surface_set` and `complex_triangulated_surface_set`.

The only difference between a tessellated face and a tessellated surface set is that the tessellated face has an additional attribute `geometric_link`. This optional attribute can be used for preserving the exact definition of the underlying exact geometry. Provided this information is used only on canonical surfaces and curves, its effect on the compactness of the STEP file is minimal. It is recommended to keep this information in order to ensure exact measurement and positioning on the tessellated entities.

Part 21 Example

```
#21=TESSELLATED_SHAPE_REPRESENTATION('',(#23),#16) ;
#23=TESSELLATED_SOLID('',(#25,#38,#43,#48,#61,#62,#63,#64,#65,#66),$
);
```

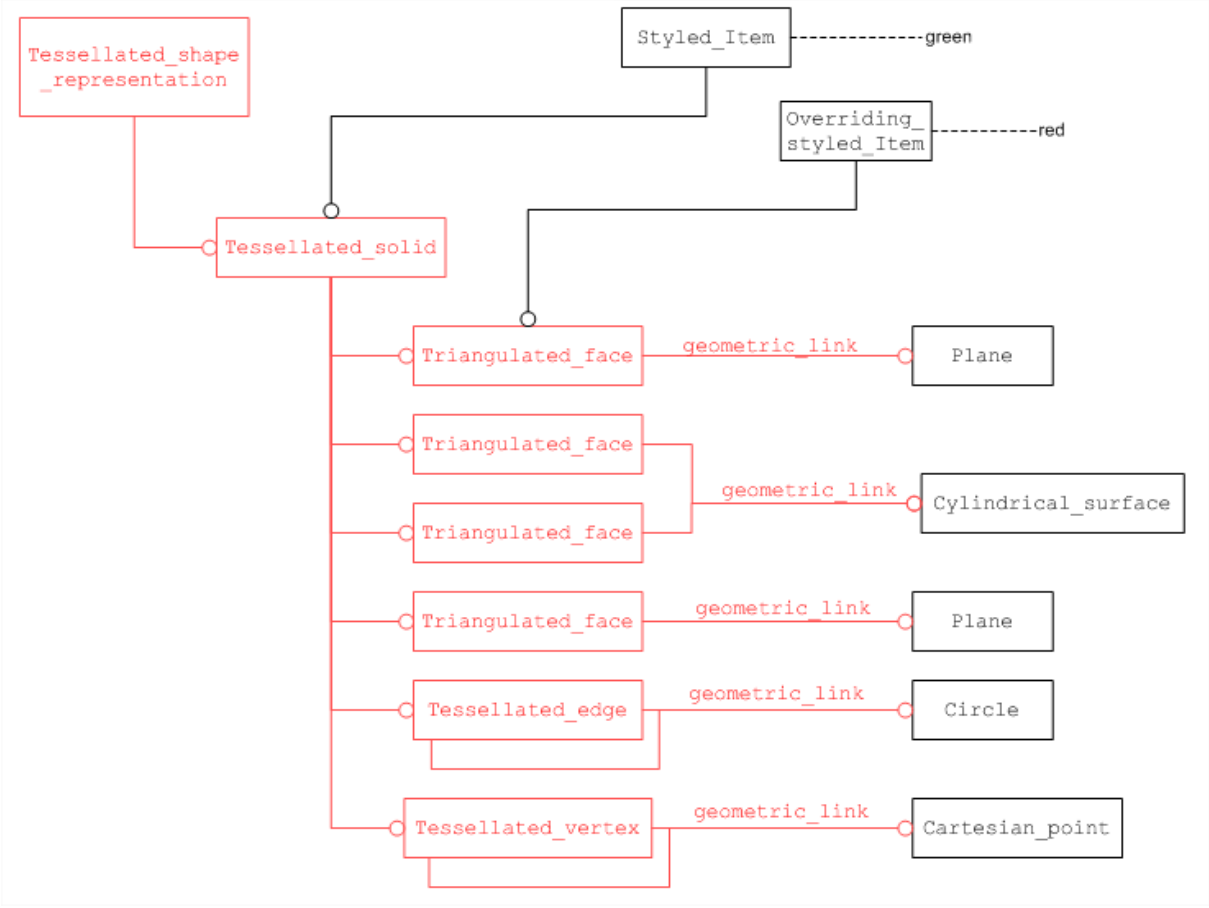

Figure 6: Structured 3D Tessellated Geometry with exact geometry information

5.4.1 Use of normals

The definition of the normals is optional for the tessellated faces in a tessellated surface set. However, when the normals are not provided, the rendering of the tessellation is poor.

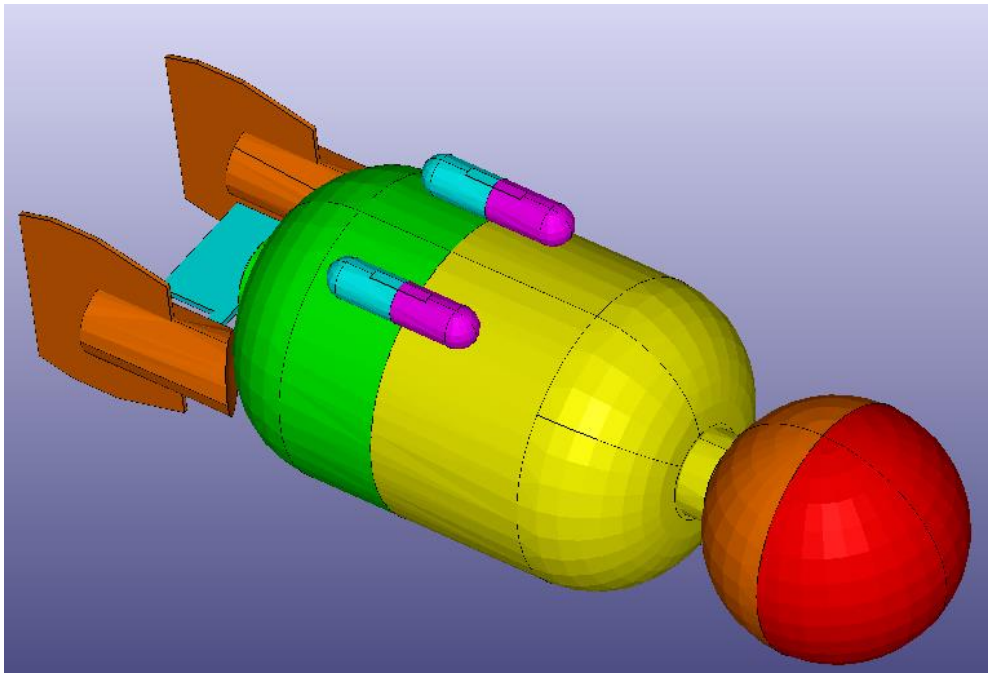

Figure 7: S1 model without normals

5.4.2 Description of a tessellated edge

A tessellated edge has an attribute coordinates referring to a `coordinates_list` entity that contains a list of points. A tessellated edge has an attribute `line_strip` defining a list of indices in the coordinates list.

```
ENTITY tessellated_edge
   SUBTYPE OF (tessellated_structured_item);
   coordinates: coordinates_list;
   geometric_link: OPTIONAL edge_or_curve;
   line_strip: LIST [2:?] OF INTEGER;
END_ENTITY;
```

A `tessellated_connecting_edge`, a subtype of tessellated edge, can be created when the tessellated edge is referenced by a `tessellated_solid` or a `tessellated_shell`. In this case, a coordinates list is no longer needed for the tessellated edge; the coordinates list of one of the faces is reused.

```
ENTITY tessellated_connecting_edge
    SUBTYPE OF (tessellated_edge);
    smooth: LOGICAL;
    face1: tessellated_face;
    face2: tessellated_face;
    line_strip_face1: LIST [2:?] OF INTEGER;
    line_strip_face2: LIST [2:?] OF INTEGER;
 WHERE
   WR1: SIZEOF(line_strip_face1) = SIZEOF(line_strip_face2);
   WR2:                              SIZEOF(line_strip_face1)                =
SIZEOF(SELF\tessellated_edge.line_strip);
 END_ENTITY;
```

Note: It is recommended to fill the attribute `line_strip` by using the indices in the `coordinates_list` of `face1` provided by the attribute `line_strip_face1`.

For a tessellated edge of the free boundary of a tessellated shell, a `tessellated_connect-ing_edge` cannot be created because the attribute `face2` is not optional. This should be improved in a later version of the Part 42.

5.4.3 Definition of tessellated faces with one coordinates_list per face

This is the simplest way to create tessellated faces. If we assume that the face is C1 continuous, there is only one normal for each point. If there is one coordinates list for a face, and one normal per point, the attribute `pnindex` is not needed.

In this case:

- The attribute `pnmax` must be equal to the size of the `coordinates_list`.
- The attribute `pnindex` must be an empty list.
- The attribute normals must have the size `pnmax` or one for a planar face.

The indices of the points defining the triangles in the attribute triangles are the indices in the `coordinates_list`.

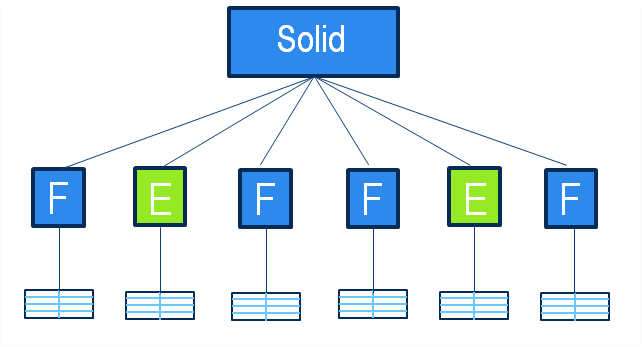

Figure 8: One coordinates_list per tessellated face

5.4.4 Definition of tessellated faces sharing a coordinates_list in a solid or a shell

It is possible to have a single `coordinates_list` for an entire solid or shell. In this case, all the faces of the solid or shell share the `coordinates_list` and all the edges as well.

- The attribute `pnmax` must be equal to the number of points representing the tessellated face.
- The attribute `pnindex` must have the size `pnmax`. It contains the indices of the points in the coordinates list.
- The attribute normals must have the size `pnmax` or one for a planar face.

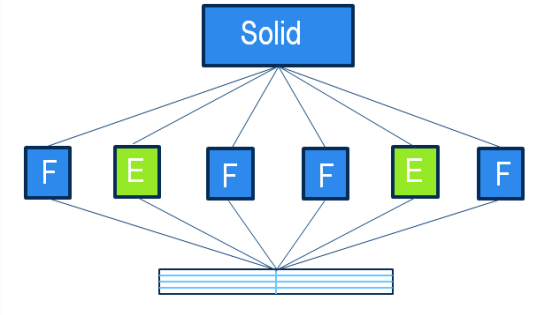

Figure 9: One single coordinates_list for a whole solid

Part 21 Example

```
#23=TESSELLATED_SOLID('',(#25,#38,#43,#48,#61,#62,#63,#64,#65,#66),$
) ;
#24=COORDINATES_LIST(' ',72,((0.,-40.,60.),(50.,-40.,60.),(0.,-
39.24039,68.68241),(50.,-39.24039,68.68241),(0.,-
36.98463,77.10101),(50.,-36.98463,77.10101),(0.,-
33.30127,85.),(50.,-33.30127,85.),(0.,-28.30222,92.13938),(50.,-
28.30222,92.13938),(0.,-22.13938,98.30222),(50.,-
22.13938,98.30222),(0.,-15.,103.3013),(50.,-15.,103.3013),(0.,-
7.101007,106.9846),(50.,-
7.101007,106.9846),(0.,1.317591,109.2404),(50.,1.317591,109.2404),(0
.,10.,110.),(50.,10.,110.),(0.,18.68241,109.2404),(50.,18.68241,109.
2404),(0.,27.10101,106.9846),(50.,27.10101,106.9846),(0.,35.,103.301
3),(50.,35.,103.3013),(0.,42.13938,98.30222),(50.,42.13938,98.30222)
,(0.,48.30222,92.13938),(50.,48.30222,92.13938),(0.,53.30127,85.),(5
0.,53.30127,85.),(0.,56.98463,77.10101),(50.,56.98463,77.10101),(0.,
59.24039,68.68241),(50.,59.24039,68.68241),(0.,60.,60.),(50.,60.,60.
),(0.,59.24039,51.31759),(50.,59.24039,51.31759),(0.,56.98463,42.898
99),(50.,56.98463,42.89899),(0.,53.30127,35.),(50.,53.30127,35.),(0.
```

```
,48.30222,27.86062),(50.,48.30222,27.86062),(0.,42.13938,21.69778),(
50.,42.13938,21.69778),(0.,35.,16.69873),(50.,35.,16.69873),(0.,27.1
0101,13.01537),(50.,27.10101,13.01537),(0.,18.68241,10.75961),(50.,1
8.68241,10.75961),(0.,10.,10.),(50.,10.,10.),(0.,1.317591,10.75961),
(50.,1.317591,10.75961),(0.,-7.101007,13.01537),(50.,-
7.101007,13.01537),(0.,-15.,16.69873),(50.,-15.,16.69873),(0.,-
22.13938,21.69778),(50.,-22.13938,21.69778),(0.,-
28.30222,27.86062),(50.,-28.30222,27.86062),(0.,-
33.30127,35.),(50.,-33.30127,35.),(0.,-36.98463,42.89899),(50.,-
36.98463,42.89899),(0.,-39.24039,51.31759),(50.,-
39.24039,51.31759))) ;
#43=COMPLEX_TRIANGULATED_FACE('',#24,36,((-1.,-0.,-
0.)),#47,(19,21,17,55,57,53,59,51,61,49,63,47,65,45,67,43,69,41,71,3
9,1,37,3,35,5,33,7,31,9,29,11,27,13,25,15,23),((1,2,3),(4,5,6,7,8,9,
10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32
,33,34,35,36,3,2)),()) ;
#44=CARTESIAN_POINT('Axis2P3D Location',(0.,0.,0.)) ;
#45=DIRECTION('Axis2P3D Direction',(0.,0.,1.)) ;
#46=AXIS2_PLACEMENT_3D('Plane Axis2P3D',#44,#45,$) ;
#47=PLANE('',#46) ;
```

5.4.5 Link between a 3D Tessellated Geometry and BREP Exact Geometry

When the 3D Tessellated Geometry and the exact BREP geometry are together in the same STEP file, it is recommended to define the link from tessellated entities to exact entities as shown in Figure 10. When reading the 3D Tessellated data:

- The access to the surfaces is indirect (it is the support entity of the exact entity).
- The access to the styling is also indirect.

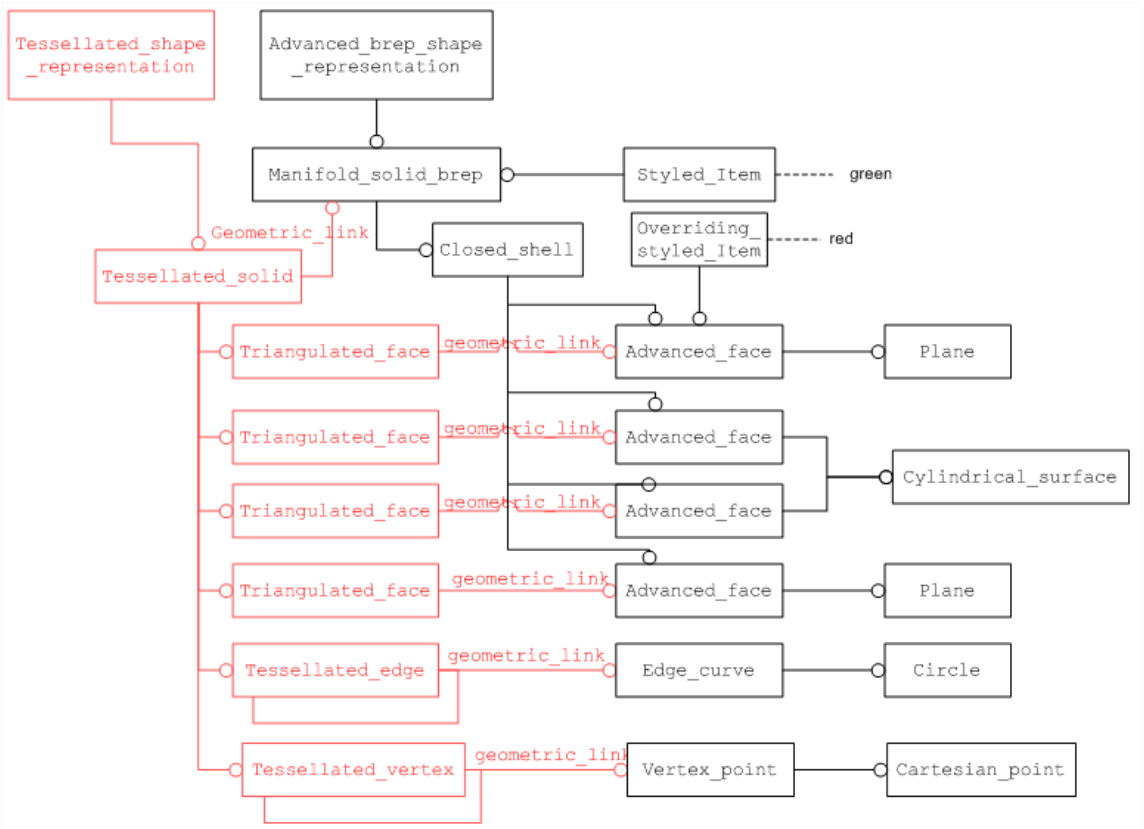

Figure 10: Structured 3D Tessellated Geometry with BREP exact geometry

6 Watertight Tessellation

In order to help ensure that the 3D Tessellated geometry is watertight, i.e. to ensure that a solid is defined without unexpected holes, it is recommended to :

- Create a single `coordinates_list` per solid (see Figure 9).
- Create a connecting edge for each edge of the solid (see Figure 11).

This is the best compromise for the size of the STEP file (no duplication of points) and for the consistency of the data.

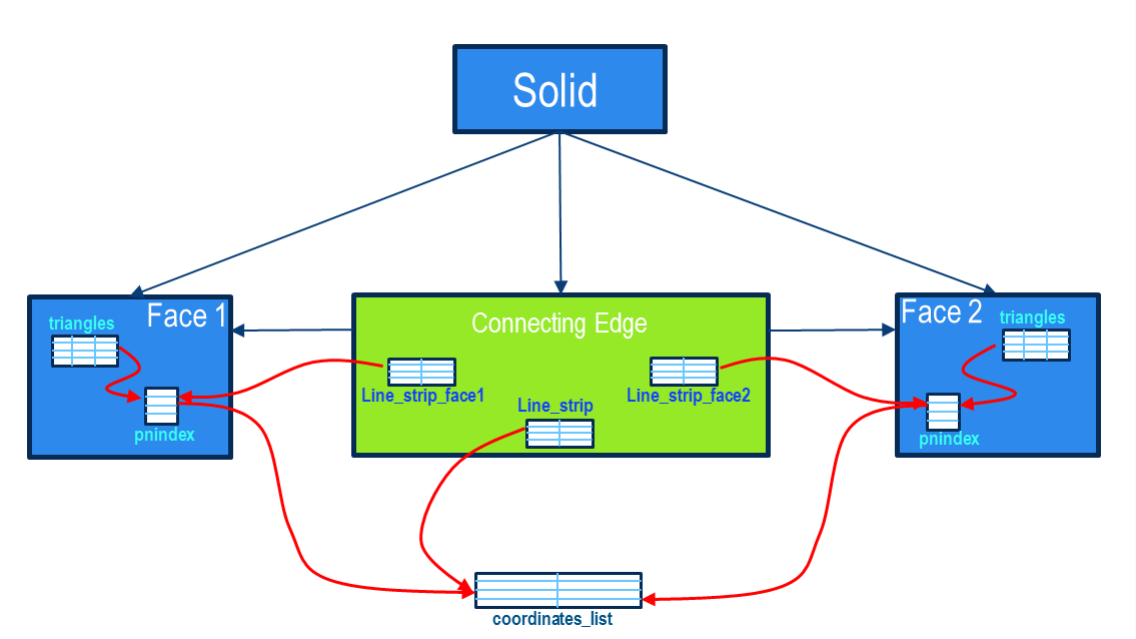

Figure 11: Watertight tessellation by using connecting edges

Note: It is possible to create connecting edges within a shell that are not boundaries of the shell. An improvement of the entity `connecting_edge` would be required to make the attributes face2 and line_strip_face2 optional.

7 3D Tessellated Geometry resulting from an assembly

The native 3D tessellated geometry is generally created from a native part. It is also sometimes created from an assembly.

As an example, an assembly design of Electrical Harness can be created by using 3D tessellated data representing the context of the plane. This context is made from parts defined by tessellated data and each Part can be the tessellation of an assembly.

In this case, the assembly structure is lost but the geometry is structured so that the geometry is not duplicated. The structure is done by using `tessellated_geometric_set` and `repositioned_tessellated_item` entities.

Note: This use case is different from the use case of an assembly having tessellated geometry that is managed in STEP as an assembly.

```
ENTITY tessellated_geometric_set
     SUBTYPE OF (tessellated_item);
      children: SET [1:?] OF tessellated_item;
END_ENTITY;
```

```
ENTITY repositioned_tessellated_item
     SUBTYPE OF(tessellated_item);
     location: axis2_placement_3d;
 WHERE
   WR1: NOT (SIZEOF (['GEOMET-
RIC_MODEL_SCHEMA.TESSELLATED_CURVE_SET',
                      'GEOMET-
RIC_MODEL_SCHEMA.TESSELLATED_GEOMETRIC_SET',
                      'GEOMETRIC_MODEL_SCHEMA.TESSELLATED_POINT_SET',
                      'GEOMET-
RIC_MODEL_SCHEMA.TESSELLATED_SURFACE_SET',
                      'GEOMETRIC_MODEL_SCHEMA.TESSELLATED_SHELL',
                      'GEOMETRIC_MODEL_SCHEMA.TESSELLATED_SOLID',
                      'GEOMETRIC_MODEL_SCHEMA.TESSELLATED_WIRE'] *
TYPEOF(SELF)) = 0);
 END_ENTITY;
```

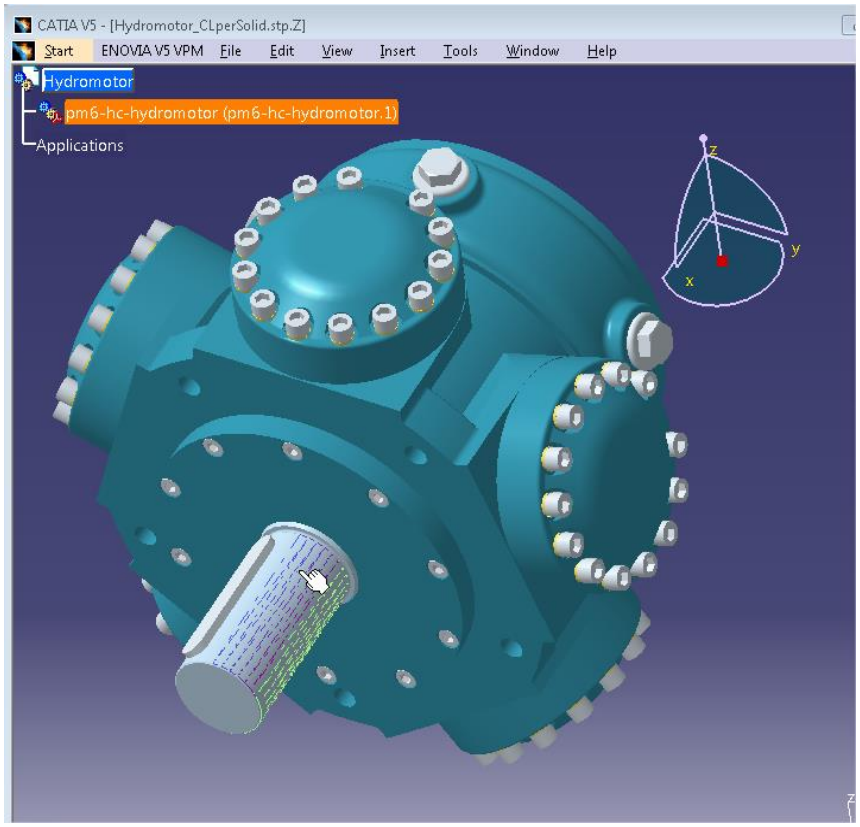

Figure 12: 3D Tessellated Geometry resulting from an assembly

Part 21 example

```
#149=(GEOMETRIC_REPRESENTATION_ITEM()REPOSITIONED_TESSELLATED_ITEM(#
145)REPRESENTATION_ITEM('')TESSELLATED_GEOMETRIC_SET((#150))TESSELLA
TED_ITEM()) ;
#145=AXIS2_PLACEMENT_3D(' ',#148,#147,#146) ;
#150=TESSELLATED_SOLID('',(#152,#161,#162,#163,#164,#165,#166,#167,#
168,#169,#170,#171,#172,#173,#174,#175,#176,#177,#178,#179,#180,#181
,#182,#183,#184,#185,#186,#187,#188,#189,#190,#191,#192,#193,#194,#1
95,#196,#197,#198,#199,#200,#201,#202,#203,#204,#205,#206,#207,#208,
#209,#210,#211,#212,#213),$) ;
```

8 Tessellated Validation Properties

Exchange of exact geometry via STEP (using BREP solids or shell-based surface models) is a well-established and stable process. One of the cornerstones for its reliability is “Geometric Validation Properties”. They allow determining the success of a data exchange by providing key values (such as volume, area, and centroid) along with the geometry data, thus enabling a comparison after translation.

This section defines and describes Validation Properties for the tessellated data in STEP. This enables validation of the tessellated geometry after exchange in the same way as it is known and relied on for exact geometry.

8.1 Fundamental Concepts

A tessellated validation property is a characteristic of a tessellated solid, surface or wireframe model, or a collection of them. When used to validate an exchange,

- The sender populates the tessellated validation properties in the exchange file, usually calculated in their CAD/geometry systems.
- The receiver performs geometric translations or transformations that are necessary on the tessellated model.
- The receiver then calculates the properties of the resultant tessellated geometry.
- The receiver compares those with the values in the exchange file.
- If they are within an agreed upon tolerance, the exchange is deemed to have been validated and successful.

The validation properties defined in this document are referred to as “tessellated validation properties”, or TVP. This is to distinguish them from the “geometric validation properties” (or GVP), which provide the same mechanism for exact BREP geometry.

8.2 Definitions of Units

The validation properties defined in this document represent different types of measures for area and length. Each requires a correct definition of the applied unit of measure in the STEP file.

A comprehensive guide on the correct definition of these and other units is given in Annex C of the CAx-IF Recommended Practices for User Defined Attributes, which can be found on the CAx-IF webpage (www.cax-if.de and www.cax-if.org) under “Joint Testing Information”.

8.3 Tessellated Validation Properties at the Part level

Tessellated Validation Properties can be attached to the tessellated geometry in a STEP file at different levels of granularity, i.e. single tessellated solids, shells or wireframes, or entire parts. Some CAx systems can determine the validation information for individual geometries within a part, whereas others are designed as a “one solid per part” system, and can only determine validation properties at the part level. The following diagram shows how validation properties are attached to the geometry defining the entire product.

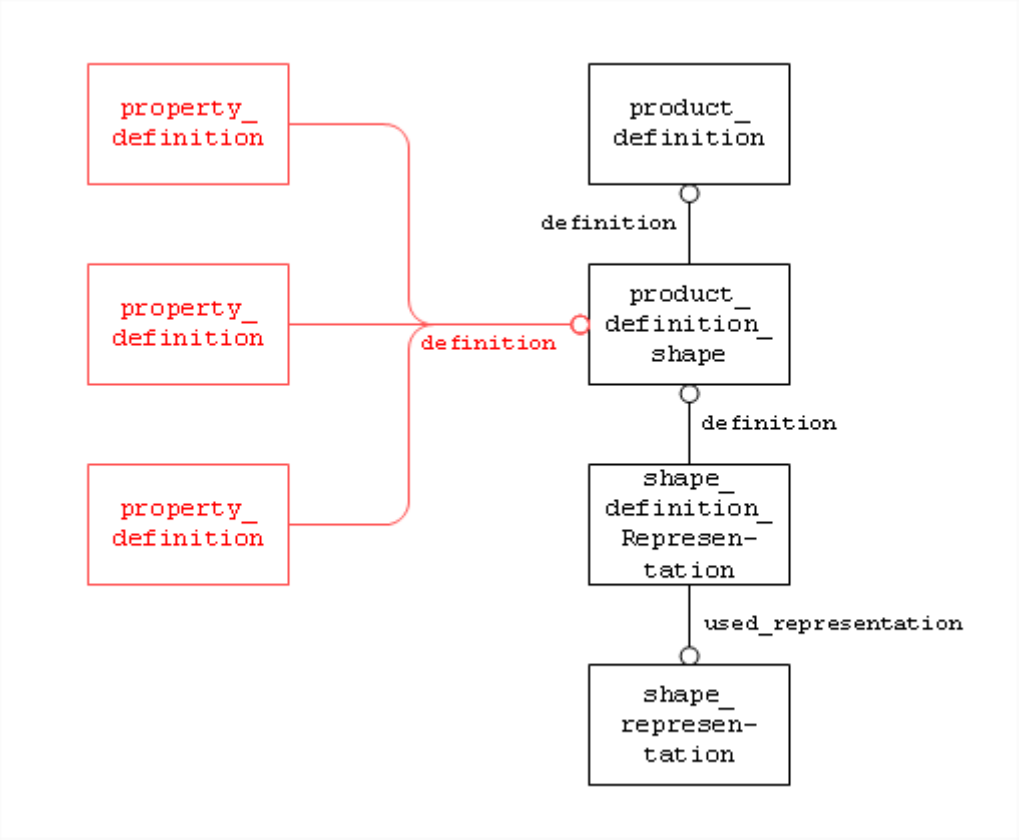

Figure 13: Tessellated Validation Properties at the Part Level

8.3.1 Bounding Box

The bounding box is a means of providing information about the model extent and location. It can be used as a further way of validating the position of the model by providing the space it fits into, in addition to the centroid. The bounding box also provides the model size, which is defined as the length of its space diagonal.

Refer to the Recommended Practices for Geometric and Assembly Validation Properties, version 4.0 or later, for the agreed approach to compute the Bounding box.

The two points $Min_{(x,y,z)}$ and $Max_{(x,y,z)}$ computed by the algorithm are stored as `cartesian_points` and define the Bounding Box Tessellated Validation Property. They provide all information needed to re-create the box as a cuboid with axes parallel to the model coordinates system and to easily determine the model size (length of the space diagonal of the bounding box = absolute three-dimensional distance between the two points).

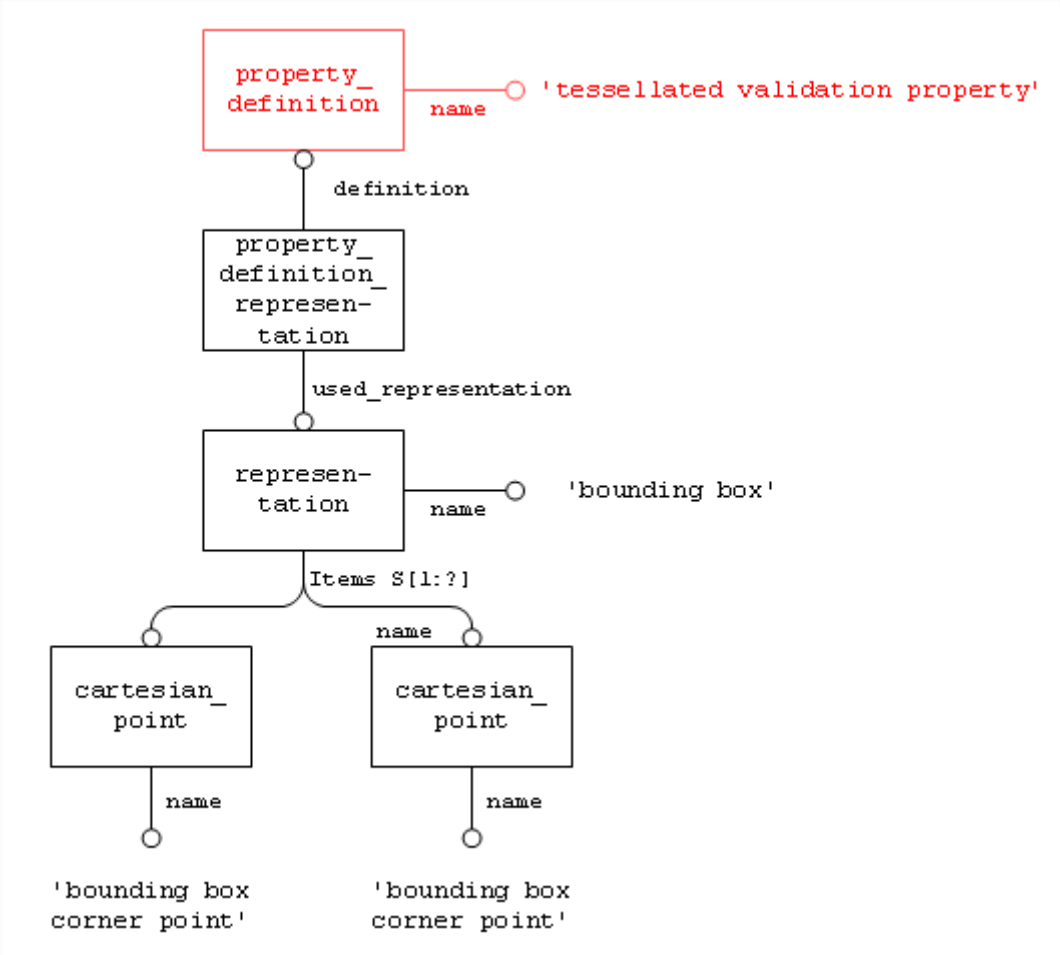

Figure 14: Tessellated Validation Property "Bounding Box"

Part21 Examples:

```
#99=CARTESIAN_POINT('bounding box corner
point',(1.79769313486E+308,1.79769313486E+308,1.79769313486E+308)) ;
#100=CARTESIAN_POINT('bounding box corner point',(-
1.79769313486E+308,-1.79769313486E+308,-1.79769313486E+308)) ;
#101=REPRESENTATION('bounding box',(#99,#100),#12) ;
#103=PROPERTY_DEFINITION_REPRESENTATION(#102,#101) ;
```

```
#102=PROPERTY_DEFINITION('tessellated validation property','bounding
box of ',#19) ;
```

8.4 Validation Properties at the Tessellated Geometry Level

Important Agreement:

Every CAD system supporting tessellated validation properties on export shall attach them at the part/product level. If an exporting CAD system also supports validation properties at the element level, it may add them additionally.

Motivation for this agreement:

- Only if the validation properties are attached at the part/product level, can it be guaranteed that every CAD system finds them. If system A only attaches them at the geometry level, and system B importing the file does not support multiple bodies per

part, it won't be able to find the information since it will look for it at the part/product level only.

- Another important motivation for this is PDM interoperability: If an assembly is exported using the "external references" mechanism, i.e. it is split into a structure file and several geometry files, the validation properties (which are deemed PDM-relevant data) will be included in the structure file only if they are attached at the part/product level. Validation properties at the geometry level are stored in the geometry files and hence are inaccessible for the PDM system.

The attachment of validation properties to a single tessellated solid, surface or wireframe within the product geometry is handled via the `shape_aspect` entity (see Recommended Practices for Geometric and Assembly Validation Properties). The recommended way to associate the `shape_aspect` with its geometry content is the way using a `geometric_item_-specific_usage (GISU)` (see Figure 15 below).

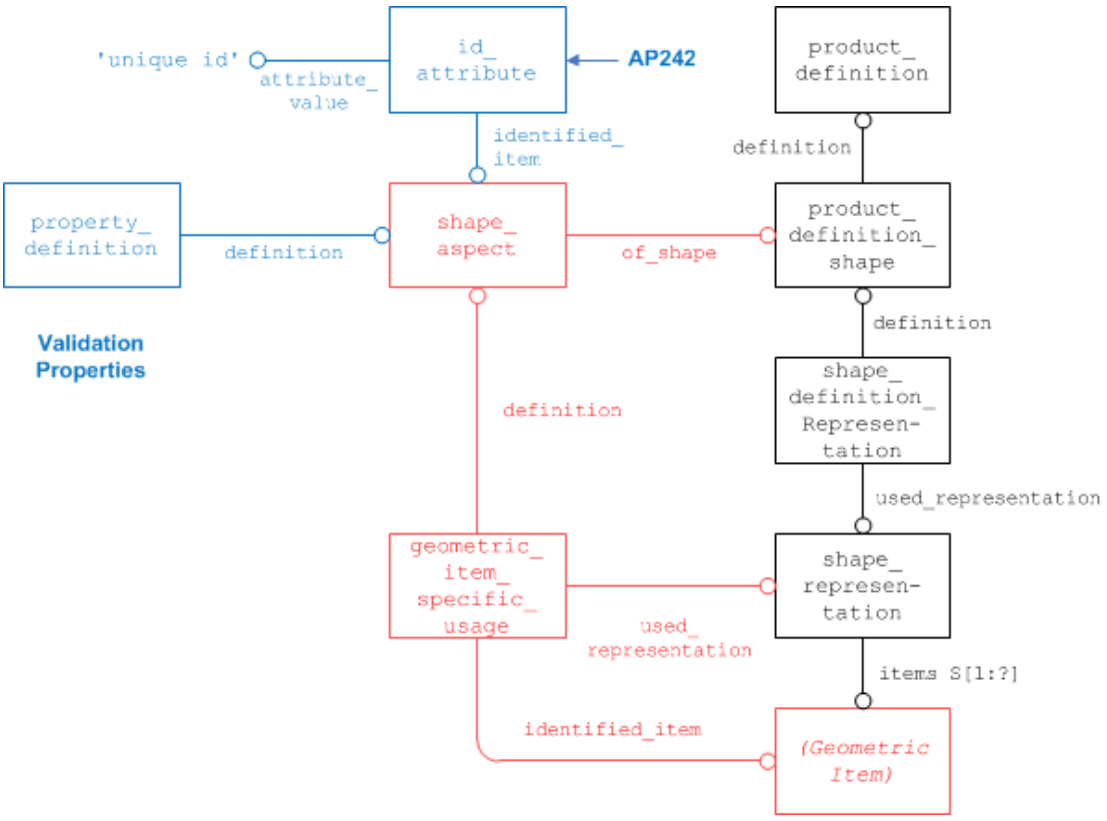

Figure 15: Validation Properties at the Tessellated Geometry level

Note: AP242 introduces a uniqueness rule on `GISU` which limits the number of `GISU` instances per `shape_aspect` to one. Since a `GISU` can relate to only a single geometric item, if several geometric elements need to be associated with a `shape_aspect`, the supertype `item_identified_representation_usage` has to be used.

In the figure above, the “Geometric Item” is the item to which the validation properties are attached. This can be a:

- `tessellated_solid`
- `tessellated_shell`
- `tessellated_surface_set`
- `tessellated_wire`

- `tessellated_curve_set`
- `tessellated_point_set`

Note: Different sets of validation properties apply to different entity types, as can be seen in the following table:

	Tessellated Geometric Entity Type	Applicable Validation Properties

3D	`tessellated_solid`
	`tessellated_shell`
	`tessellated_surface_set`
		Number of facets, tessellated surface area, tessellated surface centre point

2D	`tessellated_wire` `tessellated_curve_set`	Number of segments, tessellated curve length, tessellated curve centre point
1D	`tessellated_point_set`	Tessellated point set centre point

Table 1: Applicable Validation Properties per Tessellated Geometric Entity Type

8.4.1 Validation Properties for Solid and Surface Tessellated Geometry

8.4.1.1 Number of Facets

Any tessellated geometry consists of a number of basic geometric objects that can be connected to each other. In contrast to exact geometry with topology, in tessellated geometry the number of elements in a body is invariant, and can therefore be used as a validation property.

In the first stage of the 3D tessellated geometry capability, the number of facets equals the number of triangles.

Note: If some triangles are removed at import because they are considered as flat triangles, these triangles must be considered as found for the validation properties computation.

Figure 16: Examples of tessellated geometry for Number of Facets

Figure 17 illustrates the relevant entities and their mandatory attributes used in the assignment of a number of facets for validation.

Note: Since this capability is available in AP242 only, the entity type `integer_representation_item` is used to store the result of the count, rather than the previously used `count_measure`.

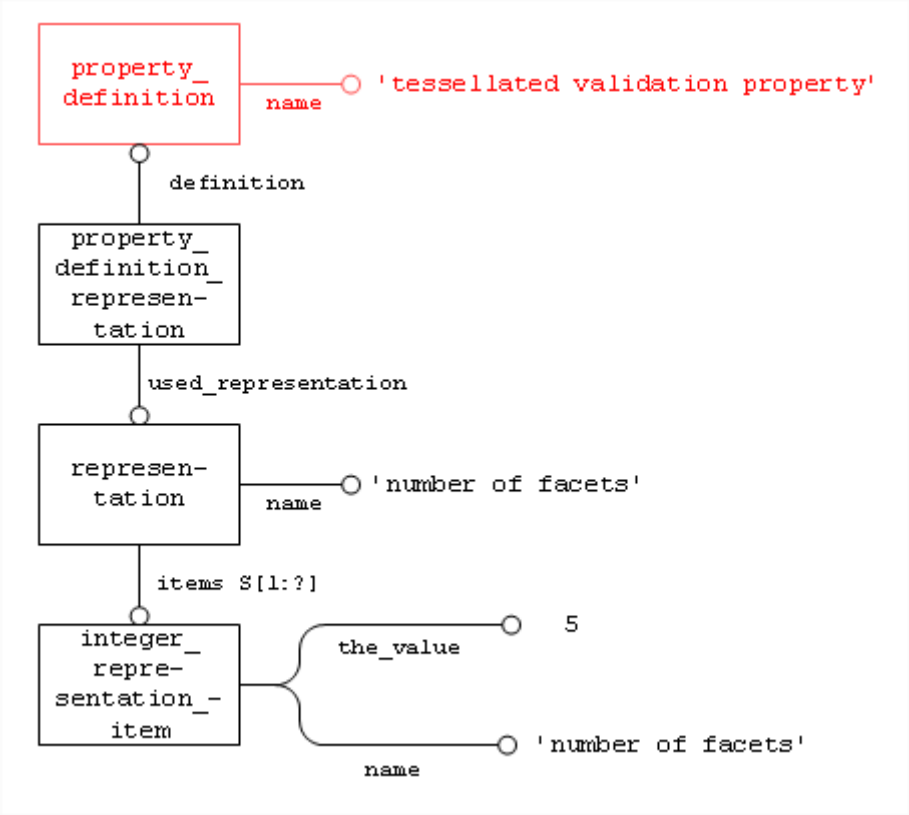

Figure 17: Tessellated Validation Property "Number of Facets"

Part21 Example:

```
#69=PROPERTY_DEFINITION('tessellated validation property','number of
facets',#53);
#70=PROPERTY_DEFINITION_REPRESENTATION(#69,#68);
#68=REPRESENTATION('number of facets',(#67),#17);
#67=INTEGER_REPRESENTATION_ITEM('number of facets',5);
```

8.4.1.2 Tessellated Surface Area

Total area specifies the area measurement of the surface of the entire tessellated solid or surface model. By default this will include any voids in the model. Figure 19 below illustrates the relevant entities and their mandatory attributes used in the assignment of the surface area validation property.

For tessellated geometry, calculation of this value is done by summing up the individual area values of all facets contained in the surface set, shell, solid or part the validation property is defined for.

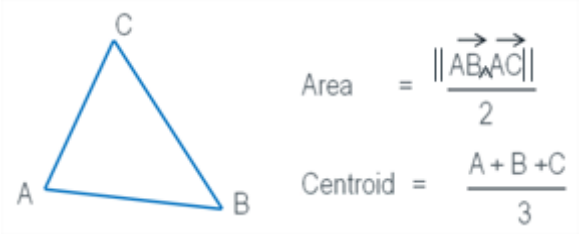

Figure 18: Calculation of Area and Centroid for a Facet

Figure 19 illustrates the relevant entities and their mandatory attributes used in the assignment of a surface area of facets for validation.

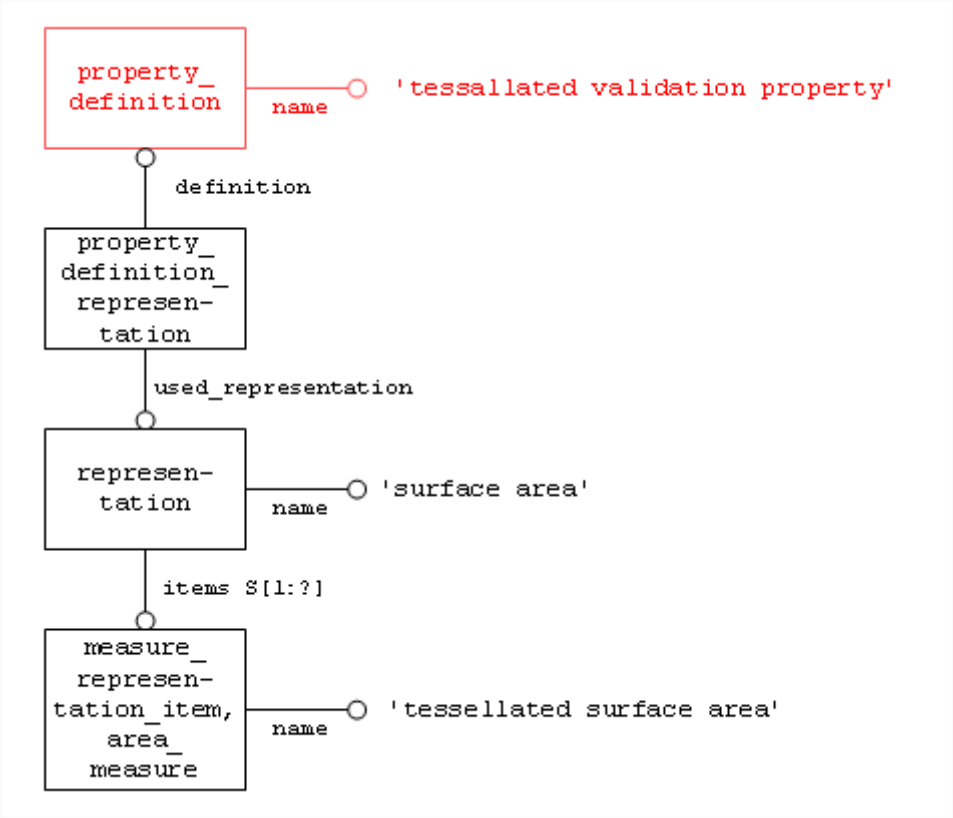

Figure 19: Tessellated Validation Property "Tessellated Surface Area"

Note: All the solids/surface facets are used in the computation so the surface area cannot be the wetted area as for Geometric Validation Properties.

Part21 Example:

```
#65=PROPERTY_DEFINITION('tessellated validation property','tessellated
surface area',#53);
#66=PROPERTY_DEFINITION_REPRESENTATION(#65,#64);
#64=REPRESENTATION('surface area',(#63),#17);
#63=MEASURE_REPRESENTATION_ITEM('tessellated surface area',
AREA_MEASURE(9600.),#62);
```

8.4.1.3 Tessellated Surface Centre Point

The centroid of Facets is the center of all tessellated entities facets in the Part. The position of the centroid is an invariant datum relative to the model origin, thus during an exchange, this can be used to validate the positional integrity of any geometric translations.

Figure 20 illustrates the relevant entities and their mandatory attributes used in the assignment of a centroid for validation.

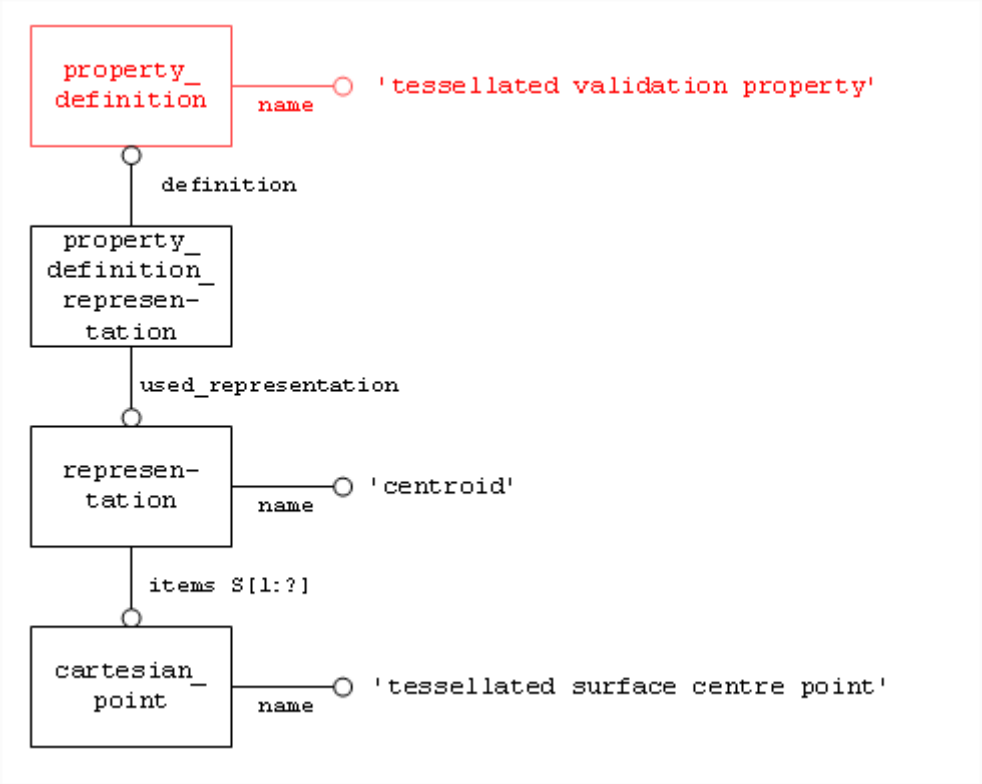

Figure 20: Tessellated Validation Property "Tessellated Surface Centre Point"

Part21 Example:

```
#59=PROPERTY_DEFINITION('tessellated validation property','centroid',#53);
#60=PROPERTY_DEFINITION_REPRESENTATION(#59,#58);
#58=REPRESENTATION('centroid',(#57),#17);
#57=CARTESIAN_POINT('tessellated surface centre point',(20.,0.,0.));
```

8.4.2 Validation Properties for Curve Tessellated Geometry

8.4.2.1 Number of Segments

Any tessellated geometry consists of a number of basic geometric objects. In the case of curves or wireframes, these are represented by a set of connected line segments. The number of segments in every curve or wireframe is invariant, and can therefore be used as a validation property.

Note: If some segments are removed at import because they are considered as null segments, these segments must be considered as found for the validation properties computation.

Figure 21 illustrates the relevant entities and their mandatory attributes used in the assignment of a number of segments for validation.

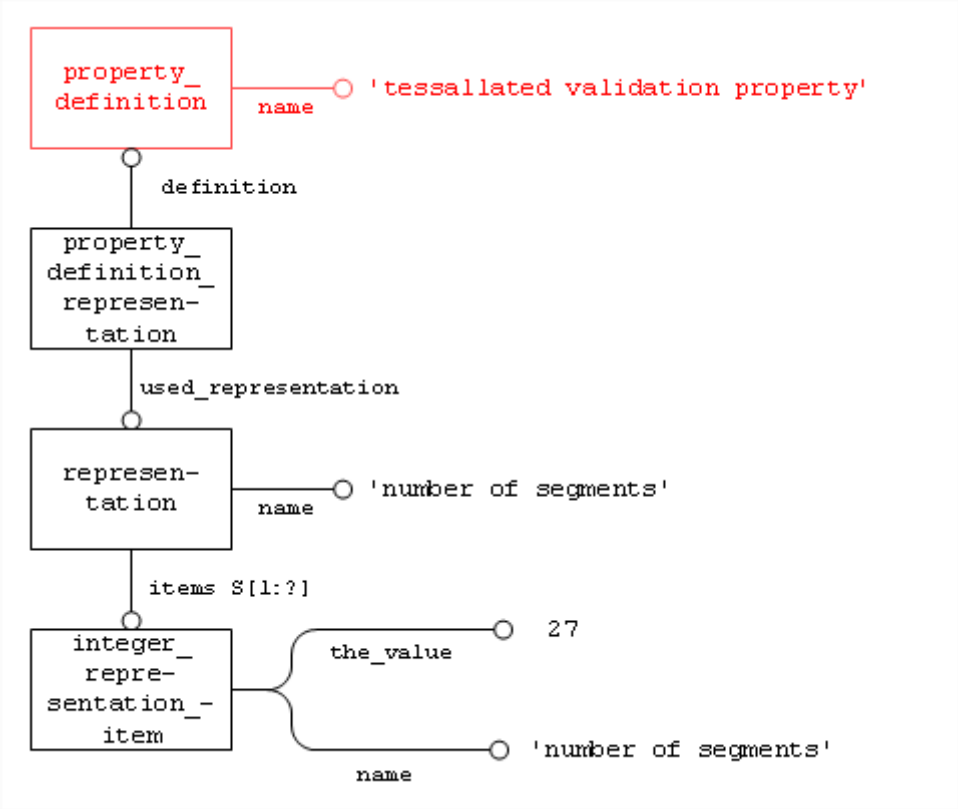

Figure 21: Tessellated Validation Property "Number of Segments"

Note: Since this capability is available in AP242 only, the entity type `integer_representation_item` is used to store the result of the count, rather than the previously used `count_measure`.

Part21 Example:

```
#117=PROPERTY_DEFINITION('tessellated validation property','number of
segments',#103);
#118=PROPERTY_DEFINITION_REPRESENTATION(#117,#116);
#116=REPRESENTATION('number of segments',(#115),#17);
#115=INTEGER_REPRESENTATION_ITEM('number of segments',27);
```

8.4.2.2 Tessellated Curve Length

A solid or surface model may contain additional, independent curves, where ‘independent’ means these curves are not edge curves of faces or solids. The length of these curves in a model can be validated to make sure the information was not lost during transfer.

Figure 22 illustrates the STEP entities required to specify the length of the independent curves in the model.

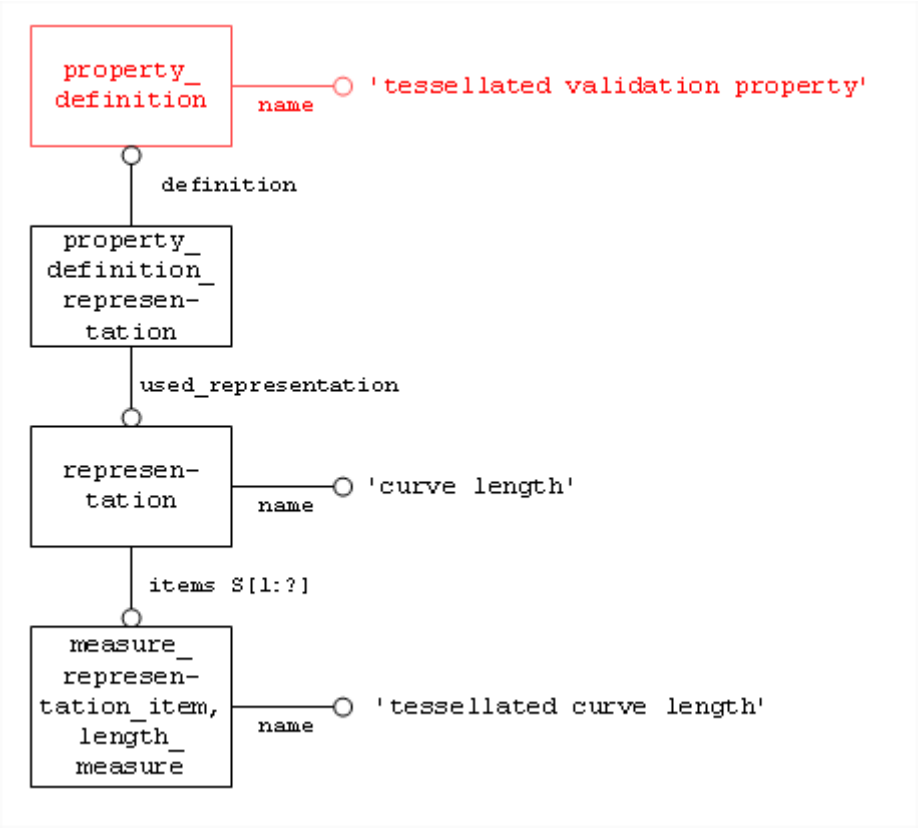

Figure 22: Tessellated Validation Property "Tessellated Curve Length"

Part21 Example:

```
#113=PROPERTY_DEFINITION('tessellated validation property','curve
length',#103);
#114=PROPERTY_DEFINITION_REPRESENTATION(#113,#112);
#112=REPRESENTATION('curve length',(#111),#17);
#111=MEASURE_REPRESENTATION_ITEM('tessellated curve length',
     LENGTH_MEASURE(249.158924051),#12);
```

8.4.2.3 Tessellated Curve Centre Point

The centroid of Segments is the center of all independent curves. The position of the centroid is an invariant datum relative to the model origin, thus during an exchange, this can be used to validate the positional integrity of any geometric translations.

Figure 23 illustrates the relevant entities and their mandatory attributes used in the assignment of a centroid for validation.

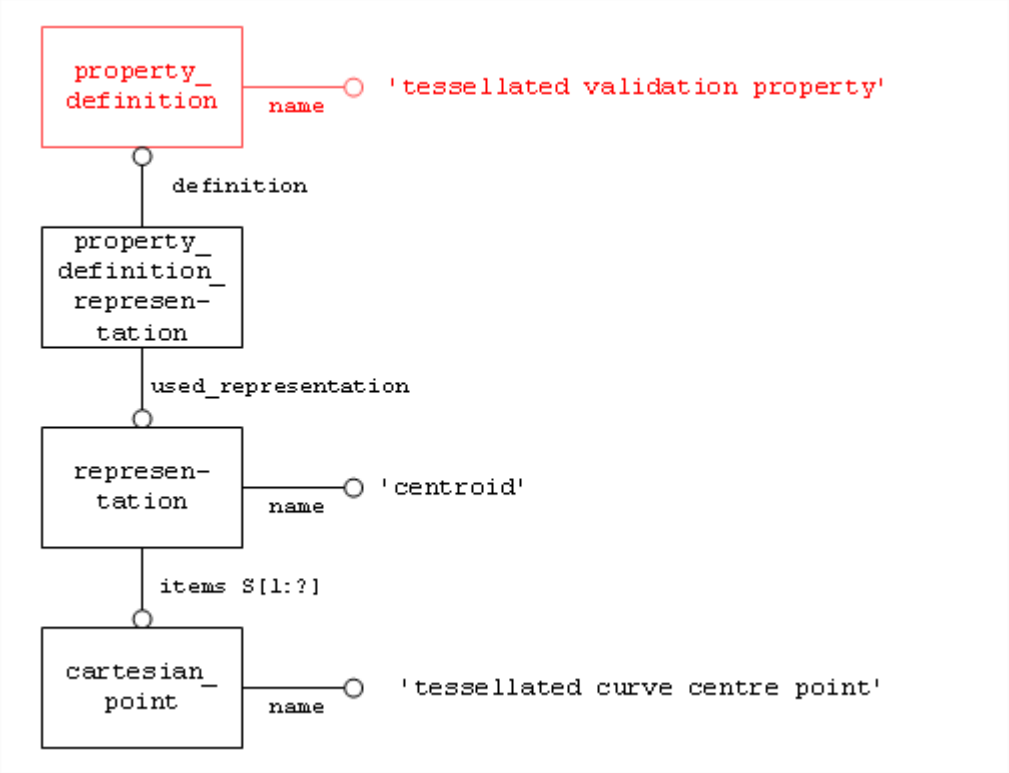

Figure 23: Tessellated Validation Property "Tessellated Curve Centre Point"

Part21 Example:

```
#59=PROPERTY_DEFINITION('tessellated validation property','centroid',#53);
#60=PROPERTY_DEFINITION_REPRESENTATION(#59,#58);
#58=REPRESENTATION('centroid',(#57),#17);
#57=CARTESIAN_POINT('tessellated curve centre point',(20.,0.,0.));
```

8.4.3 Validation Properties for Tessellated Point Set

8.4.3.1 Tessellated Point Set Centre Point

The centroid of a tessellated point set is the center of all the points it contents.

Figure 24 illustrates the relevant entities and their mandatory attributes used in the assignment of a centroid for validation.

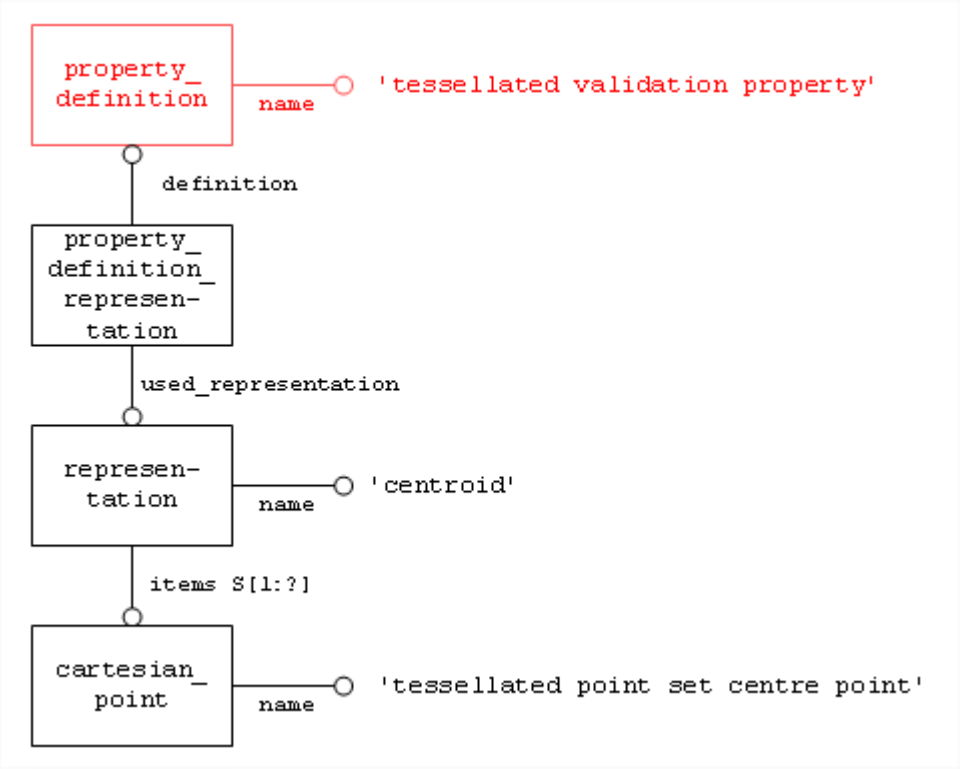

Figure 24: Tessellated validation property "tessellated point set centre point"

Part21 Example:

```
#59=PROPERTY_DEFINITION('tessellated validation property','centroid',#53);
#60=PROPERTY_DEFINITION_REPRESENTATION(#59,#58);
#58=REPRESENTATION('centroid',(#57),#17);
#57=CARTESIAN_POINT('tessellated point set centre point',(10.,50.,-20.));
```

8.5 Combining Validation Properties for Efficient Implementation

Each of the Tessellated Validation Properties described above follow the exact same pattern for its definition: `property_definition`← `property_definition_representation`→ `representation` → `representation_item`.

In a larger file, where there are many validation properties – most likely of various types – this can result in a significant number of entities, especially `representation`, which may have an impact on system performance.

Therefore it was agreed that under specific circumstances, multiple validation properties can be combined so they share the same `property_definition`, `property_definition_re-presentation` and `representation`.

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`).

In this case, the `representation.name` has to be empty. The individual validation properties will 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 ambiguity.

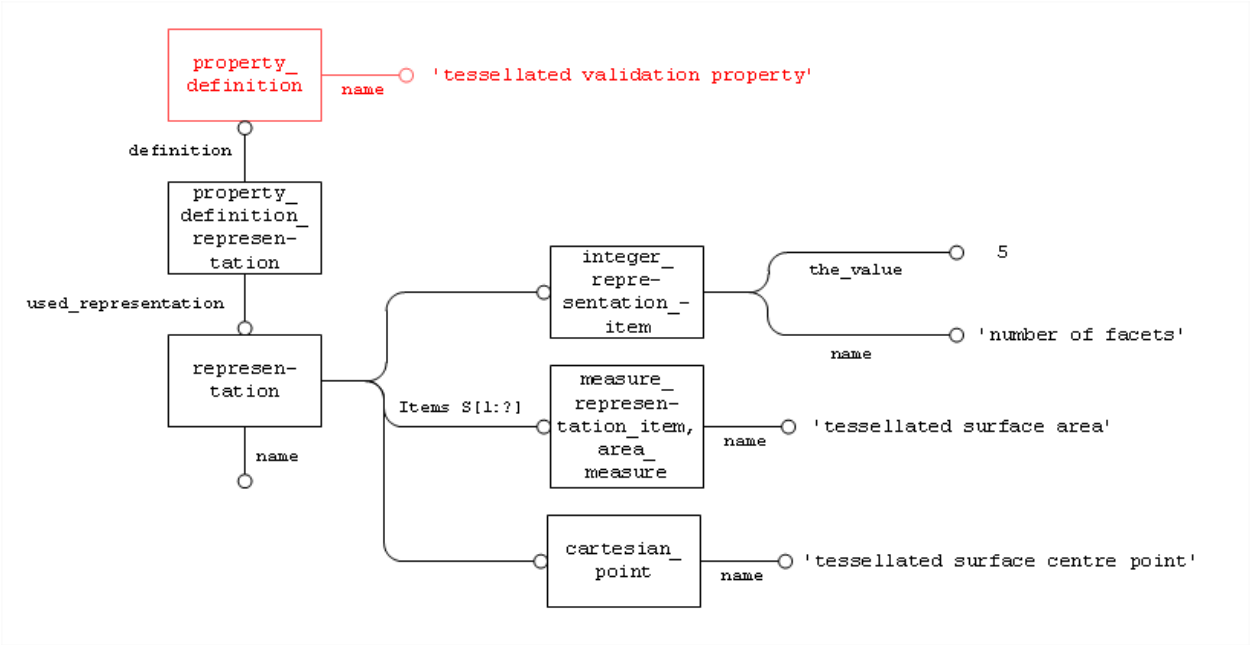

Figure 25: Combining number of facets, tessellated surface area and tessellated surface centre point

8.6 Evaluation of the Validation Properties

The differences between the geometric validation properties and the tessellated validation properties are the following:

- The exact geometric validation properties require complex computation with non-negligible computation errors.
- The tessellated validation properties require simple computation without any computation error. The errors are only numerical errors due to basic operations (addition, multiplication, division).
- The exact geometry is always managed in CAD systems by using double precision values.
- The tessellated geometry is sometimes managed in CAD systems by using simple precision values.

When the tessellated geometry is exported in STEP with the same number of decimals as the exact geometry, the accuracy of the computation of the validation properties is very high. The following thresholds can be used:

- Green result: less than 0.01%
- Yellow result: between 0.01% and 0.1%
- Red result: more than 0.1%

Depending on the number of decimals taken in the STEP file, the accuracy of the computation of the validation properties may be affected.

8.7 Assembly Validation Properties

To ensure the exchange of assemblies with parts containing tessellated geometry, it is recommended to define assembly validation properties (see Recommended Practices for Geometric Validation Properties).

8.8 Summary of Imposed Attributes Values

The following constraints on attribute values are imposed by this Recommended Practice:

Validation Property	Applicable Geometry class	PROPERTY_DEFINITION .NAME	REPRESENTATION. NAME(*)	REPRESENTATION_ITEM Subtype	REPRESENTATION_ITEM.NAME
Number of facets	Solids, Surfaces, Part	'tessellated validation property'	'number of facets'	integer _representation _item	'number of facets'
Total Surface area	Solids, Surfaces, Part	'tessellated validation property'	'surface area'	area_measure	'tessellated surface area'
Surface centroid	Solids, Surfaces, Part	'tessellated validation property'	'centroid'	cartesian_point	'tessellated surface centre point'
Number of seg-ments	Curves, Part	'tessellated validation property'	'number of seg-ments'	integer _representation _item	'number of segments'
Total Curve length	Curves, Part	'tessellated validation property'	'curve length'	length_measure	'tessellated curve length'
Curve centroid	Curves, Part	'tessellated validation property'	'centroid'	cartesian_point	'tessellated curve centre point'
Point set centroid	Points, Part	'tessellated validation property'	'centroid'	cartesian_point	'tessellated point set centre point'
Bounding Box	Part	'tessellated validation property'	'bounding box'	cartesian_point (2x)	'bounding box corner point'

Figure 26: Table of Imposed Attributes Values

(*): If several validation properties of the same type for the same model element are combined into a single property as defined in section 8.5, the `representation.name` will be an empty string.

Annex A Availability of implementation schemas

AP242

The capabilities described in this document are currently only supported by AP242.

The AP242 longform EXPRESS schema for the first edition of AP242 can be retrieved from

http://www.cax-if.de/documents/ap242_is_mim_lf_v1.36.zip