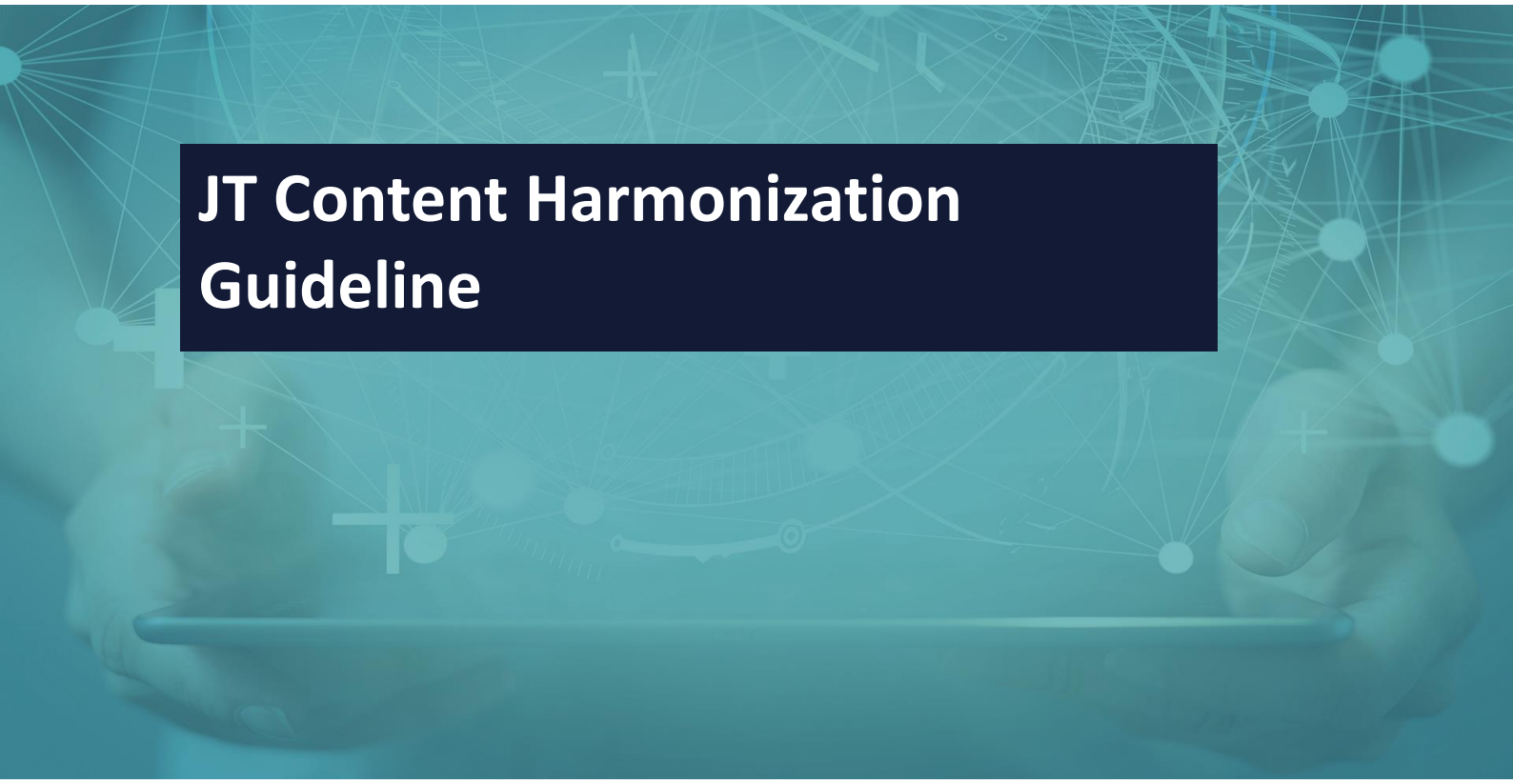


JT Content Harmonization Guideline

prostep ivip Guideline

JT Content Harmonization

Harmonization Proposal for JT and STEP AP242 XML

Version 5.1, 13 April 2022

Status: final

Abstract

In the year 2009, the ProSTEP iViP association had launched JT-activities in close cooperation with the VDA (German Association of the Automotive Industry). Within the JT Workflow Forum project group, use cases for the application of JT in the context of virtual product engineering were specified. Based on these use cases, JT-translator benchmarks were initiated to investigate the quality of converted JT-data.

For optimized process support, two action points were identified:

- P #1: So far, investigations were limited to the JT-format. However, not all requirements must address JT. For selective use cases, some of the necessary process supporting content (satisfying respective requirements), e.g. the product structure and according master data, can be provided by further formats in combination with JT**.** There is a need to analyze which contents are to be stored in JT and which contents are to be stored in accompanying formats (e.g. STEP AP242 XML). If JT and the accompanying format both provide containers for required content, there must be a unique understanding on which of the formats is responsible for the data. This is a prerequisite to application interoperability, in turn a prerequisite to the mentioned optimization of process support.
- P #2: Further, there is a need to harmonize the form, in which the above-mentioned content is to be provided.

In order to classify (JT or accompanying format) and harmonize content, it is first necessary to have a common understanding on the meaning of requirements, as both go hand in hand. In general, it can be said that contents satisfy requirements. In some cases, requirements are detailed enough to be actually synonymous to the content (see Figure 1). In others, the necessary content describes a closer look into the requirement, hence detailing it. In either case, there then exist different forms to store the required contents.

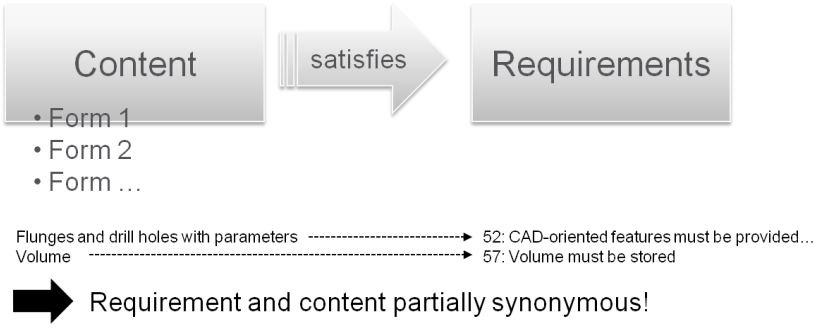


Figure 1: Correlation between requirements and content

For a prioritized subset of the JT Workflow Forum requirements, this document provides a proposal for the harmonization of contents.

Disclaimer

prostep ivip documents (PSI documents) are available for anyone to use. Anyone using these documents is responsible for ensuring that they are used correctly.

This PSI documentation gives due consideration to the prevailing state-of-the-art at the time of publication. Anyone using PSI documentations must assume responsibility for his or her actions and acts at their own risk. The prostep ivip Association and the parties involved in drawing up the PSI documentation assume no liability whatsoever.

We request that anyone encountering an error or the possibility of an incorrect interpretation when using the PSI documentations contact the prostep ivip Association (psi-issues@prostep.org) so that any errors can be rectified.

Copyright

I. All rights to this PSI documentation, in particular the right to reproduction, distribution and translation remain exclusively with the prostep ivip Association and its members.
II. The PSI documentation may be duplicated and distributed unchanged in case it is used for creating software or services.
III. It is not permitted to change or edit this PSI documentation.
IV. A notice of copyright and other restrictions of use must appear in all copies made by the user.

Content

Table of Content

Figures

1 Introduction

Based on prioritization, a proposal for harmonized content is made for a subset of requirements that were determined within the JT Workflow Forum. Requirements and respective harmonization are structured into categories, simplifying a type-based and maintainable (for future works) proposal. Within each category, this document includes the following information to each requirement:

a) A classification into “JT and/or STEP AP242 XML”
b) A description of the proposed form

The JT Content Harmonization Guideline addresses users of JT related processes and applications and improves overall data and process quality by increasing the understanding of options and possibilities provided by JT and STEP AP242 XML as accompanying format.

This document is complemented by the following documents (see Figure 2):

- The JT File Format specification, which defines the data model and file format itself. [2]
- The JT-IF Implementation Guidelines, which is aimed towards vendors with the objective to provide recommendations how to best implement certain aspects of the JT standard, by itself or in combination with accompanying formats such as STEP AP242 XML. [7]

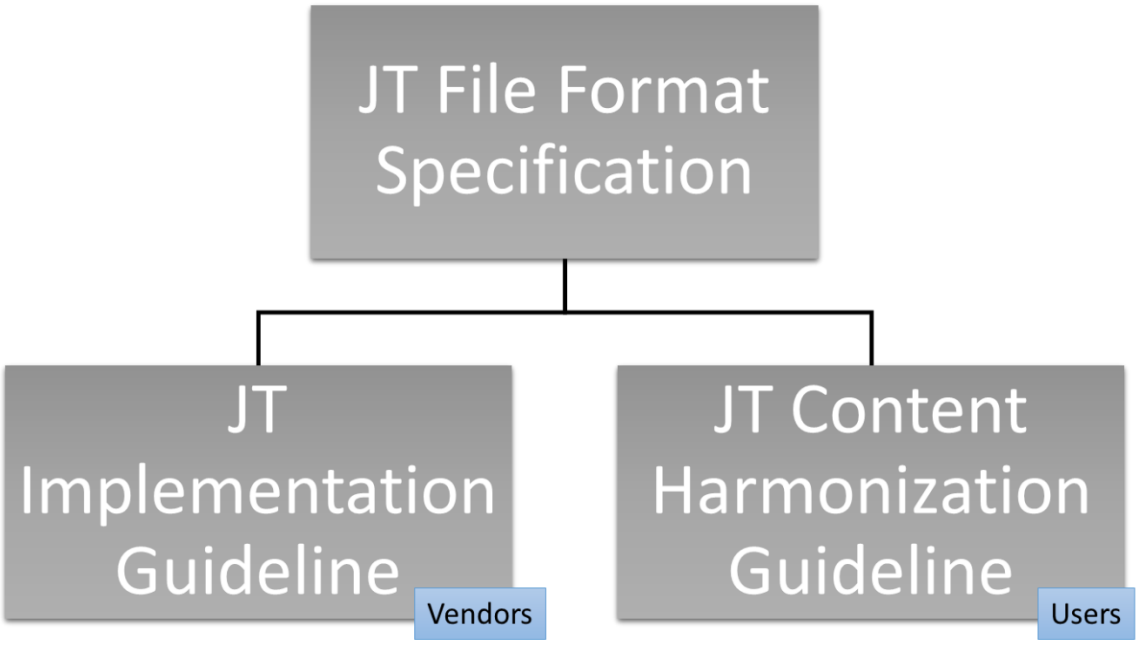


Figure 2: Documents complementing the JT ISO Specification

2 Product Structure and Globals

Harmonization of product structure refers to multiple requirements. In general, this document proposes to draw the line between JT and accompanying formats at geometry-level: The product structure, consisting of parts, assemblies and instances of both, is recommended to be mapped to the accompanying format, while the geometry of the parts (and geometry-oriented meta data) is to be stored in JT and referenced through the accompanying format's part nodes.

The primary reason for this distribution is the following: This document assumes the usage of JT and accompanying formats in the context of PDM-systems. Against this background, the content mapped to the accompanying format can easily be extracted and stored within the underlying PDM data model upon check-in, while the lightweight geometrical representation remains file-based. STEP AP242 XML is exclusively recommended as accompanying format and used interchangeably in this document.

While the current chapter describes the general default settings for the logical product structure of JT and an accompanying format, the actual physical file structure is described in chapter 8.3 of the appendix.

2.1 Assemblies and Parts

Format: JT, STEP AP242 XML

2.1.1 Requirement

All assemblies and parts must be provided (in a consistent version).

2.1.2 Description

Further information: Consistent version for parts refers to the fact that the accompanying format shall not reference some JT-files in version X (e.g. 8.1) and some in version Y (e.g. 9.5). Consistent version for assemblies is analogous, referring to a single version of the accompanying format, e.g. when the product structure is spread to multiple physical files.

2.1.3 Proposal

At current, it is recommended to use the JT version defined in PSI 14, Part 1, V 3 "JT Industrial Application Package Edition 3 JT file format specification" [2].

It is recommended to provide the product structure by a unique set of elements. Independent of the accompanying format, deprecated elements are not to be used.

2.2 Product Structure

Format: STEP AP242 XML

2.2.1 Requirement

Product structure must be stored. Instances of a part should be recognizable as such.

2.2.2 Proposal

JT is able to store assemblies, parts and instances of both. However, storage of the structure being recommended within the accompanying format implies that JT parts are not stored as instances. Such information is suggested to be within the accompanying format. The linkage from part to JT files in combination with part instances allows applications to treat instances as such.

2.3 Multi-Body Parts

Format: JT, STEP AP242 XML

2.3.1 Requirement

In CAD systems it is possible to create single parts with multiple bodies. A use case for this are single parts containing different materials or parts in different production states. It should be possible to have a similar representation in JT.

2.3.2 Proposal

This is supported in the JT format and possible to create either by creating a multibody part in the source CAD system or using a converter setting such as "convertbodynode" and "flattenbodynode" to control the structure of the resulting JT. This can be created in CATIA V5 or Siemens NX. The results show a leaf structure view that is reflected later the JT Multibody. JT multibody structure and material definition of multibody JTs should be defined in future requirements. It is also important to consider for converting back.

3 Properties

This category addresses requirements that are to be satisfied by properties. The term “properties” refers to attributes required at node level within the product structure, further describing the underlying assembly, part or instance of an assembly or part.

To describe the syntax of properties, this document makes use of regular expressions for values, allowing common means for checking the correctness. Regular expressions are formal and complex representations, consisting of normal characters (e.g. letters and numbers) and meta-characters. For more information on this matter, refer to [1].

Unless stated differently, the translator is only supposed to check the syntax of each property. A logical comparison or check is not wanted except for those properties where this proposal explicitly explains the logical meaning and how to check for it.

For property-based requirements, a list of properties or a single property is proposed, in turn described by keys and values. The key is described by a character string. For the value, a given domain is expected, described by a data type. In the following, data types are defined (following the JT file format reference where possible). Note that the floating point uses a dot (“.”) as separator. This is used to satisfy the requirement “Measurement units must be provided with a separator” and harmonize the way to do so.

List of JT property value types:

- **Floating point:**
  Regular expression for floating point with and without exponential representation
  - ^([\+\-]?[0-9]*[\.]?[0-9]+([eE][\+\-]?[0-9]+)?)$
  - ^([\+\-]?[0-9]*[\.]?[0-9]+)$

- **Integer:**
  ^([\+\-]?[0-9]*)$

- **String:**
  ^[A-Za-z0-9\.\$\^\{\ [\ (\|\)*\+\?\\\<\>_@-,;:#!'"§%&/}]~]*$

- **Length unit:**
  ^(micrometers|millimeters|decimeters|meters|centimeters|inches|feet|yards|mils|miles)$

- **Mass unit:**
  ^(micrograms|milligrams|grams|kilograms|ounces|pounds)$

- **Force unit**
  ^(newton|pascal)$

- **Force by length unit:**
  ^((newton|pascal)/(micrometers|millimeters|decimeters|meters|centimeters|inches|feet|yards|mils|miles))$

- **Layer filter**
  ^([A-Za-z0-9_]*[=][0-9]+([\,][0-9]+)*)$

- **Layer:**
  ^([0-9]+([\,][0-9]+)*)$

3.1 Length of Property Keys and Values

In JT all properties are built as key-value-pairs with the key being of data type “string” and the value being one of the above listed data types. When using property keys those are treated case sensitive which has to be taken into account when reading or writing them. According to the JT ISO file format specification the length of the string data type is restricted to 2^32 characters. In practice, the different JT translators support a property key length between 31 and 4096, and a property value length between 80 and 32000 characters. This can lead to cut off property key or value strings when using different translators.

Additionally there are restrictions for the key and value length in the authoring systems. Therefore, the lowest supported length of all keys and values in the translation tool chain should be determined. This value forms the upper bound for the length of correctly translated attributes.

3.2 String Data Type

The string data type is currently work in progress. The user requirements for the string data type are as follows:

- Full Unicode support
- Full multi-language support

Unfortunately, the current state of the art regarding JT translators does not fully support Unicode and multi-language due to technical restrictions. The capabilities vary between different CAD systems and translators which results in an unpredictable behavior when reading and writing strings. Therefore the here proposed solution describes a subset of Unicode which is known as working with all CAD systems and translators at the time of publication of this document. This results in the following regular expression as stated in the data type list above:

String: ^[A-Za-z0-9\.\$\^\{\ [\ (\|\)*\+\?\\\<\>_@-,;:#!"'§%&/}]~]*$

Expressed in simple words, right now there are usually no problems with characters that can be found on a regular English keyboard.

With future development of CAD systems and translators the goal of the Content Harmonization Guideline is to increase the allowed character set for strings successively to fully support Unicode with multiple languages.

Generally, the usage of multiple languages is not explicitly limited as long as the particular language can be represented with the proposed character set.

3.3 Mapping of Attributes

Generally, there should be a one to one (1:1) mapping of all attributes. With a CAD system as data source, all CAD attributes in assembly nodes should be translated to the accompanying STEP AP242 XML format. All CAD attributes in part nodes should be translated to the JT format.

Mapping table:

- CAD attributes in assemblies → STEP AP242 XML
- CAD attributes in parts → JT

Those attributes that are stored in JT can additionally be stored in STEP AP242 XML. Though this means a duplicate storage of the same attribute, one can benefit from the attribute in XML because attributes can already be accessed by only loading the structure.

3.4 Editing Meta Data

Format: JT

3.4.1 Requirement

Meta data must be editable in JT Translator or JT Application.

3.4.2 Proposal

JT Utilities from Siemens can edit meta data (requires license). Generic GUIs exist that help to change these attributes.

3.5 Measurement Units

Format: JT, STEP AP242 XML

3.5.1 Requirement

Measurement units must be provided per part.

3.5.2 Description

Further information: While the requirement states measurement units per part, this information is also necessary within the accompanying format, as it effects the interpretation of multiplied transformations. As any accompanying format may have its own syntax definition, a harmonization proposal is only provided for the JT property. The property is required for every single JT part. The requirement addresses the length units only. Other units are addressed by another requirement.

3.5.3 Proposal

Key **for JT property** (character string):	Value (data type)
JT_PROP_MEASUREMENT_UNITS	Length unit

When storing properties in JT, two different scenarios must be considered:

1) JT has reserved keys for default properties, called “CAD properties” in the JT file format reference [2]. These are commonly determined and in part calculated dynamically upon exporting JT from the CAD-system by the underlying translator.

2) CAD-systems provide the possibility to store user defined properties. During a JT export, these are stored in a 1:1 manner within JT. This mechanism has become widely used within industry, possibly causing various attributes to be stored multiple times with different keys.

For the property-based requirements identified by the JT Workflow Forum, this document proposes usage of user defined properties. The reason for this is that companies remain flexible in integrating batch processes and designer-responsibilities for determination of these properties, as compared to fully trusting the translator calculations. It is recommended to specify such user defined properties with a **common** prefix. This avoids overriding of standard CAD properties. The prefix “ud_” is suggested. Note that for future versions of JT and respective translators, all of the listed properties should also be provided as standard CAD properties without the prefix. In any case, it is advisable for processing applications to use the user defined properties.

As mentioned above, there exist cases in industry, in which the properties may be required to be stored within the accompanying format also. This document, however, proposes to store geometry-properties in JT at all events, leading to the requirement classification below.

3.5.4 Further Requirements

JT and STEP need to recognize units and decimal separators (point or comma). Furthermore, the Measurement Units have to be unambiguous.

3.6 Type of Unit

Format: JT

3.6.1 Requirement

Type of unit must be provided for all properties that rely on a unit.

3.6.2 Description

Further information: Any property relying on a length unit is expected to be provided in the same unit system stored within the measurement unit property (see above).

3.6.3 Proposal

Key (character string):	Value (data type)
ud_CAD_MASS_UNITS	Mass unit
ud_CAD_YOUNGS_MODULUS_UNITS	Force per area

3.7 Center of Gravity

Format: JT

3.7.1 Requirement

Center of gravity must be provided per part.

3.7.2 Description

Further information: 3 floating point properties, representing the x-, y- and z-coordinates of the position relative to the local coordinate system of the respective product structure node.

3.7.3 Proposal

Key (character string):	Value (data type)
ud_CAD_CENTER_OF_GRAVITY_X	Floating point
ud_CAD_CENTER_OF_GRAVITY_Y	Floating point
ud_CAD_CENTER_OF_GRAVITY_Z	Floating point

3.8 Moment of Inertia

Format: JT

3.8.1 Requirement

Moment of inertia must be included per part.

3.8.2 Description

Further information: 6 floating point properties, representing centroidal moments I_{xx}, I_{yy}, I_{zz} and the centroidal products I_{xy}, I_{xz}, I_{yz} of the 3x3 moment of inertia tensor. The elements I_{yx} (=I_{xy}), I_{zx} (=I_{xz}) and I_{zy} (=I_{yz}) are not to be explicitly stored.

It has to be clear which origin and coordinate system the moment of inertia is referring. Therefore, the values are calculated with respect to the center of gravity in the local coordinate system of the respective product structure node.

3.8.3 **Proposal**

Key (character string):	Value (data type)
ud_CAD_MOMENT_OF_INERTIA_XX	Floating point
ud_CAD_MOMENT_OF_INERTIA_XY	Floating point
ud_CAD_MOMENT_OF_INERTIA_XZ	Floating point
ud_CAD_MOMENT_OF_INERTIA_YY	Floating point
ud_CAD_MOMENT_OF_INERTIA_YZ	Floating point
ud_CAD_MOMENT_OF_INERTIA_ZZ	Floating point

3.9 Mass

Format: JT

3.9.1 **Requirement**

Mass must be included per part.

3.9.2 **Proposal**

Key (character string):	Value (data type)
ud_CAD_MASS	Floating point

3.10 Volume

Format: JT

3.10.1 **Requirement**

Volume must be included per part.

3.10.2 **Proposal**

Key (character string):	Value (data type)
ud_CAD_VOLUME	Floating point

3.11 Surface Area

Format: JT

3.11.1 **Requirement**

Surface area must be included for parts.

3.11.2 Proposal

Key (character string):	Value (data type)
ud_CAD_SURFACE_AREA	Floating point

3.12 Material Thickness

Format: JT

3.12.1 Requirement

Material thickness must be included per part. Units for thickness must be same as those in JT Properties.

3.12.2 Proposal

Already exists as:

Key (character string):	Value (data type)
ud_CAD_PROP_MATERIAL_THICKNESS	Floating point

3.13 Critical Parts

Format: JT

3.13.1 Requirement

Critical parts, e.g. heat or cooling sources, must be recognizable as such.

3.13.2 Description

Critical parts should be marked as such with the two JT Properties described here. The CRITICAL_TYPE property contains the type of the critical part while the CRITICAL_VALUE property holds the value for the type.

3.13.3 Proposal

Key **for JT property** (character string):	Value (data type)
CRITICAL_TYPE	String
CRITICAL_VALUE	Floating Point

3.14 Translator Settings as Property

Translators offer numerous settings to configure the translation process according to the user's demand. This includes common JT configuration settings like those stated in Appendix 8.2 but also vendor specific settings that may be unique for the respective translator. For an improved documentation and traceability, it can be useful to store the translator settings that were used to create a JT file. This can be done with an extra file or right inside the created JT file in a property.

Storing translator settings is optional and the used technique is not harmonized. Below one can find a proposal how to store the settings in a JT property. However, due to the highly variable nature of translator settings, the created property will usually only be useful in conjunction with the originally used translator.

Informative proposal for an optional property for translator settings:

Key **for JT property** (character string):	Value (data type)
UD_TRANSLATOR_SETTINGS	String

Potential information that could be included in such a property:

- Source file / source system
- Translator name with version
- General JT settings
- Translator specific settings
- STEP AP242 XML settings
- Etc.

3.15 Data Quality Check Seal as Property

Format: JT

3.15.1 **Requirement**

Data quality check seal must be stored e.g. as attribute.

3.15.2 **Proposal**

Suggestion to define an attribute for:

- check-tool attribute,
- check-profile
- check-result attribute with various levels.

These attributes can be defined like the JT-Translator attribute (see 3.14).

Quality checkers are still very specific, so the attributes could remain check-tool and profile specific.

A further recommendation is to use a JT Image stamp.

3.16 Link from JT to External File

Format: JT ISO 14306:2016

3.16.1 **Requirement**

Link to "external" material (files in the file system/data base/ PDM system) in JT files should be possible.

3.16.2 **Proposal**

Proposal given in chapter 7.8.3 of Implementation Guideline [7].

4 Geometry

This category addresses requirements that relate to the geometry itself, being further divided into the sub-categories “exact geometry”, “tessellated geometry”, “reference geometry” and “geometry healing”.

4.1 Exact Geometry

Format: JT

4.1.1 Requirement

Exact XT B-Rep data must be stored.

4.1.2 Description

Translators must provide all geometrical and topological information.

4.1.3 Proposal

JT provides means to store exact geometry in three different ways: JT B-Rep, XT B-Rep and primitive geometry, such as cylinders, boxes, etc.

It is suggested to use **XT B-Rep** geometry to satisfy the above requirement.

4.2 Tessellated Geometry with LODs

Format: JT

Harmonization of tessellated geometry representations ensures more homogeneous processes. Particularly, processes that rely on large assemblies, possible from multiple internal or external responsibilities, call for holistically unique visualization and geometry-based functionality, such as collision detection within Digital Mock-Up analyses. Respectively, as can be seen below, a requirement stated the necessity for predefined levels of detail (LODs).

In JT, LOD settings are valid and equal for the entire part geometry, i.e. any kind of edge or curve is tessellated with the same parameters independent from their use, position or logical function. Feature suppression forms an exception here. For further information on tessellation parameters, refer to Appendix 8.

This document makes a suggestion for three predefined, user-defined LODs with a respective name that is expected. Their **existence is optional** to further allow company-specific handling of JT configurations, but the proposal can be considered good practice for the mentioned application scenarios.

The following suggestion holds: If a LOD, with one of the here listed names, is provided in JT, then it must be created with the here specified parameters.

4.2.1 Requirement

Tessellated geometry is included with predefined LODs.

4.2.2 **Description**

- Three user defined levels of detail with names and parameters
- Presence is optional
- If present and named according to the recommendation, the parameters are mandatory
- Levels are not defined by actual numbers to allow further LODs to be inserted
- The included LODs are sorted so that finer tessellation parameters must be represented by decreasing level, as suggested in the JT file format reference [2]

4.2.3 **Proposal**

Name	Chordal	Label	Angular	Length	Feature Suppression	Simplify	AdvCompression-Level
ud_FINE	0.2	ud_FINE	45	0.0	0	1.0	0.0
ud_MEDIUM	0.8	ud_MEDIUM	45	0.0	0	1.0	0.0
ud_COARSE	1.8	ud_COARSE	45	0.0	0	0.15	0.0

The proposed LODs have by experience shown to satisfyingly cover the following application fields:

LOD “ud_FINE”: This LOD reflects a fine tessellation for a relatively precise approximation of the original geometry. For example, it can serve in DMU-processes for a detailed analysis.

LOD “ud_MEDIUM”: This LOD reflects a compromise between the other two. It has a reduced accuracy and is proposed for VR processes, where large component sizes need to be processed at performant visualization frame rates.

LOD “ud_COARSE”: This LOD reflects a coarse tessellation for high-performance visualization scenarios. It should be used by visualization-systems for the effective movement of parts and assemblies.

The following is valid for all three proposed LODs: Features are not to be suppressed, nor is an absolute chordal length to be assumed and specified. To ensure accuracy of an approximation and a faster processing of the JT content, the compression level should set to 0.0. The angular parameter is to be set to 45, since empirically this represents the optimal balance between performance and the quality of the tessellation.

It is proposed to set “advCompressionLevel” to a value of 0.0. The reason for this setting is the intolerance in error within follow-up processes that use the JT. Further, the suggested LODs are to be specified without simplification, since a reliable approximation could not be guaranteed otherwise.

For ud_FINE and ud_MEDIUM the “Simplify” value should be set to 1.0 because a value other than 1.0 can override the chordal and angular settings. This could lead to unpredictable results in the tessellation quality which is not welcome. If a simplification or reduction of size is required this should be accomplished by changing chordal and angular values of either the existing LODs or additional LODs. For ud_COARSE the “Simplify” value should be set to 0.15 to achieve a significant reduction of size and details. Because this coarse LOD is not used for quality relevant topics, the unpredictable result is acceptable since the focus is more on good performance than predictable quality.

For all LODs, exceeding the above proposal, the following condition holds: The included LODs are sorted so that finer tessellation parameters must be represented by decreasing level, as suggested in the JT file format reference [2].

In addition to the LOD-specific parameters, the global value for “chordalOption” is suggested to always be set to ABSOLUTE. Only then a reliable approximation can be guaranteed for follow-up processes.

See Appendix 8.2 for an example config file.

For editing the LOD, Teamcenter Visualization and JT-Utilities are suggested.

4.3 Relationship between Exact and Tessellated Geometry

Format: JT ISO 14306:2016, STEP AP242 XML

Linking information between exact and tessellated geometry.

4.3.1 Requirement

It should be possible to store Properties/meta information on the level of construction faces (XT BRep) and the related tessellated data representation.

4.3.2 Proposal

Reference to specification in chapter 5.7.2 of Implementation Guideline [7].

4.4 Bounding Boxes

Format: JT, STEP AP242 XML

4.4.1 Requirement

Bounding boxes must be included.

4.4.2 Proposal

Bounding boxes must be included on part level in any case.

On assembly level, the bounding boxes have to be stored in the accompanying format.

4.5 Supplemental Geometry

According to the JT Implementation Guideline, there are various synonymous terms for supplemental geometry, depending on the application context and the CAD system used [7]

- supplemental geometry
- construction (constructive) geometry
- auxiliary geometry
- design geometry
- support geometry
- cosmetics
- reference geometry

Supplemental geometry is stored using PMI (Product Manufacturing Information). Refer to Chapter 0. The term *supplemental geometry* will replace the terms reference geometry, auxiliary geometry etc. to create a uniform wording between different workgroups.

4.6 CAE Features

Format JT, STEP AP242

4.6.1 Requirement

To speed up simulation process (e.g. holes, drill-holes, muffs) CAE features should be included in JT.

4.6.2 **Proposal**

Originally, it was proposed to store this supplemental geometry in JT below the JTKPart node. Since then, it has been proposed to reference this geometry out of a STEP AP242 XML. The geometry mentioned here is located sub-part below the JTKPart node.

4.7 Geometry Healing

Geometry healing is highly depending on use cases and involved applications. Therefore, it is not possible to harmonize this topic with settings or recommendations that are generally valid. JT itself does not contain any healing functionality so any kind of healing is always translator specific and may vary for different products and vendors.

Below one can find some information regarding healing that may be helpful. Though this information comes from the JT Implementor Forum it is not considered as harmonized or generally valid but just informative:

- From a JT Toolkit perspective, there are no inherent healing functions made available.
- Healing, if applied, is always preferred to be done on the CAD side, rather than the JT side.
- The prerequisite for all healing capabilities is a corresponding checking mechanism to detect flaws in the model. Which checks are done is a user decision depending on the use case (selection from larger list).
- How the defects are healed exactly varies from tool to tool.
- Whenever healing is applied, it means that the original model geometry is modified. It depends on the use case to which extent these modifications are acceptable.
- The prerequisite for any healing is a checking mechanism which detects the defects that need healing
    - The quality criteria which are typically being used for checking are those defined by SASIG PDQ and / or VDA 4955-2.
    - Hence, further discussion of healing capabilities should always be based on these criteria.

- Levels of healing applied:
    - Some translators do not heal, but report model issues in the log file
    - Some tools provide basic automated healing capabilities, which are switched on by default, but can be disabled by the user
    - Some tools provide additional advanced healing capabilities; some automatic, some interactive.
    - Some tools also offer to compare the original model with the derived geometry to illustrate deviations
- The primary goal of all healing tools is to create a valid closed solid.

- The „most prominent“ geometry defects that are being checked (and healed):
    - Tiny Elements:
        - Short boundary segments, small faces
        - Elements with an extent very close to, or even below, the model accuracy
        - Collapsing boundaries/faces will cause topology issues
    - Large Gaps:
        - Distances between neighboring faces, or underlying surfaces and their respective edge curves, close to or even above the model accuracy
        - Will cause loss of faces or unclosed solids
    - Trimming Problems:
        - Especially for faces / edges with spherical (toroidal) / circular definition
        - May cause “donuts” to appear instead of rounded edges
    - Other:
        - Self-intersections
        - Sharp edges

4.8 Transparent Texture

Format: JT

4.8.1 Requirement

Support of transparent textures in JT -files (internal/external) must be possible.

4.8.2 Proposal

The transparency of a texture can be handled with BlendType and BlendColor, see JT File Specification [2] chapter 6.2.2. The definition of textures in JT is similar to OpenGL – the implementation itself inside a JT application depends on the used graphic libraries and is not a topic of the JT file format.

The texture image data can be stored in different ways (see Table 30 - Texture Vers-1 Image Format Description Pixel Format values [2]), among these are also formats supporting the alpha channel (opacity) and therefore allows transparent textures.

4.9 External Texture Files

Format: JT

4.9.1 Requirement

Texture files from file system, database or PDM system

4.9.2 Proposal

The image data of a texture can be stored either inside the JT document or externally, i.e. driven by a flag for the texture and – in case of external storage – the name of external location. It is possible to store in JT but more often used externally. It is recommended to store this information in an external format.

Three different versions of texture format are documented in the ISO File Specification [2].

For “version 1” of texture data storage in JT, the corresponding elements are:

- Flag for storage internal / external: “U8 : Inline Image Storage Flag”.
- Storage name: “MbString : External Storage Name”.

For versions 2 and 3 of the texture format, corresponding similar definitions exist.

The external storage name is only present if data field “Inline Image Storage Flag” equals “0.” If present, there will be an instance of the “External Storage Name” for each image. This string is a relative path based name for the texture image file. Where “relative path” means the string contains the file name along with any additional path information that locates the texture image file relative to the location of the referencing JT file.

Any other way of external texture location – such as data base related storage - will be an individual solution and will require a specific implementation.

As a fallback mechanism for a missing referenced texture file, the application can use the material definition if present.

A further proposal for implementation is to define an identifier in the JT that refers to an external texture databank.

4.10 Threads

Format: STEP AP242 XML, JT

4.10.1 Requirement

Hole and screw threads must be identifiable by users and applications. Must have the same functionality as in CAD. For instance, in the CAD system one can define where the thread is based on surface and hole diameter.

4.10.2 Proposal

Currently there is no large demand for a geometrical representation of threads in JT, i.e. a 3D helix representation. Recommendation to use PMI (preferably semantic) instead of geometrical representation (Implementation Guideline chapter 7.5.3) [7]:

- Thread information shall be transferred at least as graphic PMI annotation
- As far as supported by the exporting translator, the PMI data given for the thread shall be enhanced to be valid semantic PMI information in the JT file. These semantic PMIs must be human and machine readable.
- Until there is dedicated support for supplemental geometry in JT, geometric representations of threads (e.g. cylindrical surfaces, helices) shall not be exported.

There must still be a possibility to derive the thread for a 2D drawing from the JT, as not all are working in a drawingless process. This could be achieved by extracting construction geometry created in the source system. In Catia V5 for instance, the geometry such as the profile and the circular head can be extracted from the symbolic thread and exported as curves in the JT.

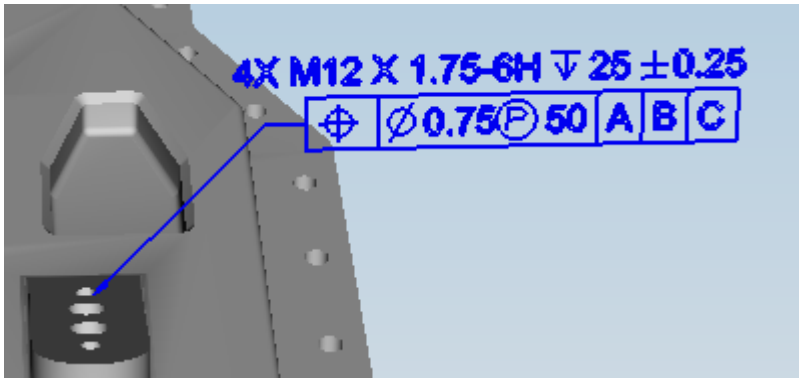


Figure 3: Example for Thread Information (4X M12) in a PMI Annotation [7]

The support of semantic thread information in JT ISO is an important and ongoing discussion there is currently no consensus for implementation.

Short-term solution: It is best to separate product and supplemental geometry into separate nodes and to have a parent node with PMI annotations that reference entities in both of them. When the native CAD model is a part rather than an assembly, the best practice is to define the top-level node as a part node (rather than an assembly) and the nodes below it as SUBNODES (rather than assembly components).

4.11 External Links to JT Elements

Format: STEP AP242 XML, JT

4.11.1 Requirement

Especially for kinematic simulations, it has to be possible to reference elements of the JT geometry from other files (e.g. STEP AP 242 XML). It must be stable and possible to change references e.g. during change management.

4.11.2 Proposal

One important mechanism for identification of JT elements is the Moniker ID (see Chapter 6.2. Implementation Guideline) [7]. This Moniker ID is already defined in JT. An example of the implementation with moniker ID is shown in the following section.

Further proposals include creating a B-Rep ID and a Face-ID. A Face-ID should be used for surface and tolerance analysis.

After further discussion in the Workflow forum it has been determined that, not only a Moniker ID would suffice but also a Face ID will be needed.

4.11.3 **Pointer Syntax Example**

Using a Moniker-ID the link is set from the XML assembly file, the following syntax can be used:

The pointer specification defines a generic syntax with which to specify fragment identifiers for entities contained in non-XML file types. A fragment identifier structure is derived by examining the relationships between the entities in the file.

Example fragment identifier structure for content in the XT segment of a JT file:

Identifier	Entity represented
/	root
/XT-doc	document element
/XT-doc/body	body
/XT-doc/body/face	face

Using this identifier set, the following types of references could be defined.

Fragment identifier syntax	Description
xml(XT-doc/body[@name='Crankshaft'])	Refers to any bodies called "Crankshaft" in the XT file, using the SDL/TYSA_NAME Parasolid attribute.
xml(XT-doc/body/face[@id='42'])	Refers to the faces in the XT file with id=42, using the Parasolid identifier as if it was an attribute (XT_ENTITY_ID).

4.11.4 **Validation of External Links**

To check if an external link to JT elements exists and is consistent, the following validation criteria have been proposed:

1) Do Face IDs or Moniker IDs exist in the JT
2) Do the ID values remain consistent in the source CAD and JT

Without the first criteria, there can be no certainty that an external link exists. The second criteria makes sure that the links remain between the same elements. If the IDs change, one can only visually check if the external references have remained the same.

5 Product Manufacturing Information

Generally the PMIs (Product Manufacturing Information) containing strings should follow the restrictions for the string data type described in chapter 2.3, 3.1 and 3.2 to guarantee readability and correctness of the created PMIs. This is particularly important for semantic PMIs and their machine-readable portion.

5.1 PMI Data

Format: JT

5.1.1 Requirement

PMI data should be stored in the JT file.

5.1.2 Proposal

PMI can be stored in their respective PMI containers or as attributes. JT Open Toolkit provides an API to read or write this information

5.2 PMI Font

Format: JT

5.2.1 Requirement

Font of PMI must be supported. Symbols need to be correct.

5.2.2 Description

PMI and annotation need to have a property defining its font (identified by the translator in the source CAD and to be used in the JT application visualizing the JT content).

5.2.3 Proposal

One should consider declaring a PMI font. Otherwise, it could happen that problems occur during conversion. The Font information is stored in PMI properties; see ISO File Specification (14.8.2 PMI Properties). There is different font-related information stored.

Note: Some systems do not read the font from the JT files and use some default views instead.

5.3 Tolerances

Format: JT, STEP AP242 XML

5.3.1 Requirement

Tolerances must be included.

5.3.2 Proposal

Tolerances on part level must be placed in JT.

Assembly level tolerances must be placed in STEP AP242 XML.

5.4 Supplemental Geometry

Format: JT

5.4.1 Requirement

Supplemental (Auxiliary) geometry must be included and identified as such.

5.4.2 Proposal

Because STEP has a list of items considered supplemental geometry, the group suggests to adapt this list.

It was stated that the currently used applications are not able to perform measuring operations between PMI objects and geometry objects. This should be taken into consideration for further action regarding the categorization of supplemental geometry.

As there was no general consensus on this requirement, further investigation is needed.

Generally, supplemental geometry should be stored in a layer. A final proposal regarding layers has to be created.

Some current suggestions to identify supplemental geometry as discussed in the JT CH Group include:

- to place supplemental geometry in a layer and convert using layer settings
- through NoShow settings in conversion
- through reference sets
- converting supplemental geometry to a separate file

5.5 Section Planes

Format: JT, STEP AP242 XML

5.5.1 Requirement

Section planes must be included.

5.5.2 Proposal

Section planes on part level are stored in JT using PMI technology.

Section planes on assembly level are stored within STEP AP242 XML.

5.6 Links to Features

Format: JT

5.6.1 Requirement

Technological information must be linked to geometrical features.

5.6.2 Proposal

This is supported through PMIs.

5.7 Section Views

Format: JT

5.7.1 Requirement

Creation and storing of section views must be possible.

5.7.2 Proposal

This is possible for normal section views after defining in source CAD and conversion. Problems still exist with special offset section views.

5.8 Assembly level PMI

Format: JT, STEP AP242 XML

5.8.1 Requirement

It must be possible to place PMI on assembly level with references to multiple parts.

5.8.2 Description

There are several constellations to have PMI inside an assembly structure. These are the following:

1. Model View on assembly root level referencing PMI on the same level

2. Model View on assembly root level referencing PMI on a lower sub-ordinated assembly level (sub-assembly PMI)

3. Model View on assembly root level referencing PMI on a lower sub-ordinated part level (part PMI)

4. Model View on sub-assembly level referencing PMI on the same sub-assembly level (similar to case 1)

5. Model View on sub-assembly level referencing PMI on a lower sub-ordinated assembly level (sub-sub-assembly PMI) (similar to case 2)

6. Model View on sub-assembly level referencing PMI on a lower sub-ordinated part level (part PMI) (similar to case 3)

7. Model View on part level referencing PMI on its own part level (part PMI)

8. Model View with PMI mixed from the cases 1 – 7

5.8.3 Proposal

PMI on assembly level is already implemented in JT. However, is has to be checked if all different cases can be applied.

6 Various

6.1 Textures

Format: JT

6.1.1 Requirement

Textures must be included.

6.1.2 Proposal

The agreement is that two versions are valid and can be used as required:

- Textures included directly in the JT file
- Textures with a reference to an external texture object

6.2 Colors

Format: JT

6.2.1 Requirement

Colors must be stored for entire part or geometrical features.

6.2.2 Proposal

Colors can be set per part and per shape with the JT Open Toolkit.

It is recommended to use the latest JT Open Toolkit to set the shape materials and add their RGBA color portion to the respective XT Entities.

To avoid performance problems a material should be set per shape and face but not per JtkPart.

6.3 Color Legend

Format: JT

6.3.1 Requirement

Color legend has to be stored.

6.3.2 Proposal

Deformed vertices are colored. A legend describes the meaning of the colors.

Colors can be set per part and per shape with the JT Open Toolkit.

6.4 Digital Rights Management

Format: JT, STEP AP242 XML

6.4.1 Requirement

Digital rights management data must be included.

6.4.2 Proposal

Currently there are two different ways of integrating DRM and JT into the workflow:

1. One can use a container-based technique comparable to an encrypted archive file. This is a less process invasive solution where the JT file is placed inside an encrypted file.
2. One can integrate a proprietary DRM inside the applications. This way the applications are capable of decrypting JT files encrypted with the specific proprietary encryption.

The second solution is preferred and this requirement is rather application oriented than a matter of proper harmonization.

6.5 Systematic Arrangement of Documents

Format: JT

6.5.1 Requirement

Must fulfill the DIN 6789 "Systematic arrangement of documents".

6.5.2 Proposal

The DRM process is used to support this requirement. See above requirement.

6.6 Measurement Values Separator

Format: JT

6.6.1 Requirement

Measurement values must be provided with a separator.

6.6.2 Proposal

In STEP AP242 XML always a dot is used as separator for measurement values. This is recommended for JT as well.

6.7 Reference Point System

Format: JT

6.7.1 Requirement

Reference Point System (RPS) information must be stored in JT.

6.7.2 Proposal

RPS is supported through viewing applications and supplemental geometry.

6.8 Coordinate System

Format: JT

6.8.1 Requirement

Coordinate systems must be provided in JT.

6.8.2 Proposal

Coordinate systems are supported in JT and can be used as required.

6.9 Electronic Signature

Format: JT, STEP AP242 XML

6.9.1 Requirement

Electronic signature necessary for unique identifiability (to be checked by Legal dept.) in context with release process.

6.9.2 Proposal

This requirement should be covered by the PDM/PLM system or the archiving system.

6.10 Mechanism and Motion

Format: STEP AP242 XML, JT

6.10.1 Requirement

This requirement can be summarized in the following points:

- Kinematic data, fixes, joints, joint attributes kinematical links must be stored.
- Kinematical bearings including degrees of freedom must be stored.
- In a hybrid assembly model, it must be possible to use input from JT data as well as native CAD data for creating kinematic simulations. (It is a long-term goal of the JT WF to store kinematic data in STEP AP242 XML.)
- An animation is stored either as key frame animation or with all needed frames.

6.10.2 Proposal

The recommendation is to use complementary formats such as STEP AP242 XML XML for the kinematic definition and animation sequences. Doing so, the JT files can be referenced and used as visualization of the parts. This is described in the recommended practices for kinematics in STEP AP242 XML for reference [9].

The following are further arguments why STEP AP242 XML should be used for mechanism and motion:

- STEP AP242 XML can be used for the kinematic definition and JT for the geometry.
- The Kinematic_pair concept in STEP AP242 XML provides a sophisticated method to define constraints, position of joints and degrees of freedom.
- Kinematical joints can now be transported/converted via STEP AP242 XML, the background is that this is then compacter. Kinematics are defined together in the assembly file with position and joints.

Storing kinematic information and animation scenes within a JT file directly, i.e. by using JT properties, would be technically feasible but would require reading the JT properties of each involved JT file before setting up the scene in an application. The JT format itself provides the geometrical representation, which can be used for animation scenes. Using complementary files also helps to reuse the JT files in other processes.

6.11 Multi-Body Simulation

Format: JT

6.11.1 Requirement

For multibody simulation, the following must be identifiable as such:

- Flexible items (with routings)

- Fixings (handles)
- Connectors

6.11.2 Proposal

Supported in JT via attributes.

6.12 Definition of Material Variants

Format: JT

6.12.1 Requirement

For an alternative display of geometry objects, it should be possible to define variants, as for the switching between day - and night design, for an existing/assigned material. A further use case is for calculating center of gravities for multiple configurations.

6.12.2 Proposal

The current format specification provides the ability to define materials as documented above. For a given node in the logical scene graph (i.e. JT Part) it is one definition. To be able to switch between different materials by user request, the current solution would be to double the 3D design and assign a different material for it. In this case, the common layer or reference set methodology (see Section 3.3 of the Implementation Guideline [7]) can be used to switch on / off the different states.

The disadvantage of this approach is the bigger amount of data, as the LOD and geometry data will not be shared between the different states. It is very important that the right body is measured in the end. This is currently just a workaround; if the demand increases, a better solution is needed. One suggestion is using STEP AP242 XML.

6.13 Material Identification

Format: IAP 2

6.13.1 Requirement

Material names must be included in JT and special materials must be identifiable as such.

6.13.2 Proposal

The JT Open reader writer supports this. A whitepaper to this topic was released in 2011.

6.14 Managing different states of parts

Format: JT, STEP AP242 XML

6.14.1 Requirement

It should be possible to manage different states of parts.

6.14.2 Proposal

See Chapter 2.3 Multi-Body Parts.

For managing different states of parts, it is possible to use layer handling, reference sets or Model Views. The recommendation for the future is, to use only the reference sets. (see Implementation Guideline Chapter 3.3.)

6.15 Viewing on Multiple Operating Systems

Format: JT

6.15.1 **Requirement**

JT should be viewable on multiple operating systems.

6.15.2 **Proposal**

Various viewers are available for multiple platforms, including mobile devices.

7 References

[1] John E. Hopcroft, Jeffry D. Ullman: Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie. Addison Wesley, Bonn 1994, ISBN 3-89319-744-3.

[2] PSI 14, Part 1, V 3 JT Industrial Application Package Edition 3 JT file format specification, July 2021

[3] ISO 10303-242:2014, Industrial automation systems and integration -- Product data representation and exchange -- Part 242: Application protocol: Managed model-based 3D engineering. 2014.

[4] Siemens PLM Software: Getting Started with the JT Open Toolkit. February 2013.

[5] Siemens PLM Software: JT Open Toolkit Functional Description. February 2013.

[6] CAx Implementor Forum: Recommended Practices for AP242 Business Object Model XML Assembly Structure, Release 3.00. November 2021.

[7] JT Implementor Forum, Implementation Guidelines, Version 3.0. December 2018.

[8] VDA Empfehlung 4961/3 - Abstimmung der Datenlogistik in SE-Projekten, Anhang SE-Checkliste. April 2012.

[9] CAx Implementor Forum: Recommended Practices for STEP AP242 Edition 2 Minor Revision Domain Model XML Kinematics, Release 1.00. November 2021.

8 Appendix

8.1 LOD-Parameters

LODs are defined in a JT-configuration file with the following attributes. However, theoretically not all of the mentioned attributes must be specified.

- **level:** **Indication of the level-of-detail**
  Integer: The finer the approximation, the lower the number should be.
- **chordal:** **Tolerated chordal (edge) deviation**
  Distance between the linear approximation and the exact curve or surface: If 0 is specified, this parameter is ignored in the tessellation.
- **angular:** **Tolerated angular deviation**
  Angle (measured in degrees) between sequential segments of the approximation: If 0 is specified, this parameter is ignored in the tessellation.
- **length:** **Absolute length of the segment-tendons**
  If specified, the parameters angular and chordal are ignored; each edge has the specified length.
- **advCompressionLevel:** **compression level of the LOD**
  Any value between 0.0 (no compression) and 1.0 (highest compression): This affects the use of different CODECs. Higher compression can lead to inaccuracies in the approximation!
- **simplify** **Simplification relative to the first LOD (level)**
  Any value between 0.0 (0%) and 1.0 (100%), referring to the number of polygons. If the specified percentage is not achieved by the initial tessellation, the meshes are further simplified with this attribute.
- **featureSuppression** **Minimum size in hole and arch features**
  Holes and arches smaller than the specified value are ignored in the tessellation.

In addition to the above options, which are specified per LOD, there exists another parameter that has a strong impact on the tessellation of all LODs.

- **chordalOption** **Meaning of the chordal parameter (edge deviation)**
  Either ABSOLUTE or RELATIVE: ABSOLUTE implies that the chordal parameter specifies an absolute value with respect to the underlying unit of measurement. This leads to the fact that larger parts are represented by more tessellation than smaller parts. RELATIVE, on the other hand, implies that the chordal parameter represents a percentage with respect to the underlying part size.

8.2 Example config file

chordalOption = "ABSOLUTE"

writeWhichFiles = "ALL"

pmiOption = "PART_AND_ASM"

compression = true

advCompression = true

advCompressionLevel = 0.0

triStripOpt = true

seamSewing = false

seamSewingTol = 0.001

```
includeBrep = true
brepPrecision = "DOUBLE"
autoNameSanitize = false
updateChangedPartsOnly = false
writeAsciiAssembly = false
singlePartsNoAssem = false
autoLowLODgeneration = false
smartLODgeneration = true
numLODs = 3
JtFileFormat = "105"

LOD "1" {
        Level = 1
        Chordal = 0.2
        Label = "ud_FINE"
        Angular = 45
        Length = 0.0
        FeatureSuppression = 0
        Simplify = 1.0
        advCompressionLevel = 0.0
}

LOD "2" {
        Level = 2
        Chordal = 0.8
        Label = "ud_MEDIUM"
        Angular = 45
        Length = 0.0
        FeatureSuppression = 0
        Simplify = 1.0
        advCompressionLevel = 0.0
}

LOD "3" {
        Level = 3
        Chordal = 1.8
        Label = "ud_COARSE"
        Angular = 45
        Length = 0.0
        FeatureSuppression = 0
```

Simplify = 0.15

advCompressionLevel = 0.0

}

8.3 JT File Structures

The recommended product structure for JT is described in Chapter 0 where the recommendation is based on the assumption of the presence of a PDM system. This leads to the segmentation of geometry (JT) and structure information (STEP AP242 XML) which is motivated and justified in detail in Chapter 0.

While Chapter 0 handles only the theoretical concept of the product structure, the current chapter covers the actual file structure and how it is created by translators.

Generally the drawbacks of JT+STEP AP242 XML are easily outweighed by the numerous advantages.

Though some of the use cases utilizing JT benefit from special file structures which are occasionally superior to the default recommendation of JT and STEP AP242 XML. This is usually explained by a lack or change of assumed prerequisites, e.g. PDM context, unlimited bandwidth, small file size etc.

To account for these special use cases the current chapter illustrates special requirements of use cases regarding file structures.

8.3.1 Use cases

The following use cases are considered:

- 3D-Measurement and –Analysis
- Archiving
- Bidding / Inquiry
- CAE Data Visualization
- Digital Factory Building Planning
- Digital Factory Manufacturing Planning
- Digital Factory Material Handling
- Digital Factory Plant Development
- Drawingless Manufacturing
- ECAD/MCAD Collaboration
- Factory DMU
- Finite Element Analysis (FEA/FEM)
- High-end Visualization
- Hybrid Design in Context
- Identification for Location Based Viewing
- Identification of Part/Assembly
- Installation Feasibility
- Material Specification
- Multibody Simulation
- Multimedia Annotations
- Non-hybrid Design in Context
- Packaging
- Pressline Simulation
- Prototyping Processes
- Supplier Integration (OEM to Supplier)
- Supplier Integration (Supplier to OEM)
- Systems Engineering
- Tolerance Studies
- Viewing
- Viewing on mobile devices in the pre-series

8.3.2 **File structures**

The following file structures are considered [4][6]:

- JT only - Per part: All assembly nodes in a product structure hierarchy are stored in a single JT file, and each part node in the hierarchy is stored in an individual JT file in a subdirectory that is of the same name as the assembly JT file. The subdirectories are created on export if they do not already exist.
- JT only - Fully shattered: Each product structure node in the hierarchy is stored in an individual JT file.
- JT only - Monolithic: All product structure is stored in a single JT file.
- JT only - Mimic: Siemens PLM Software JT translators typically define one common custom mapping, Mimic, in which output JT file content matches input CAD content on a one-to-one, file-to-file basis.
- STEP AP242 XML + JT - Nested Structure
- STEP AP242 XML + JT - Nested Structure with additional part-level XML files

Additionally it is possible to create custom and mixed file structures where each node can be saved in any type of structure. See Figure 7 and Figure 8 for examples.

The following example illustrates each of the three standard mappings. Given the following JT Open Toolkit product structure hierarchy, where A1, A2, and A3 are JtkAssembly nodes; P1, P2, and P3 are JtkPart nodes; and P1' is a JtkInstance of P1 [4]:

```
A1
+-A2
| +-P1
| +-P2
+-A3
+-P1'
+-P3
```

The standard mappings would result in the following JT files:

Mapping	Files Generated
Per part	A1.jt A1/P1.jt A1/P2.jt A1/P3.jt
Fully shattered	A1.jt A2.jt A3.jt P1.jt P2.jt P3.jt
Monolithic	A1.jt

The JT Open Toolkit Functional Description provides some examples [5]:

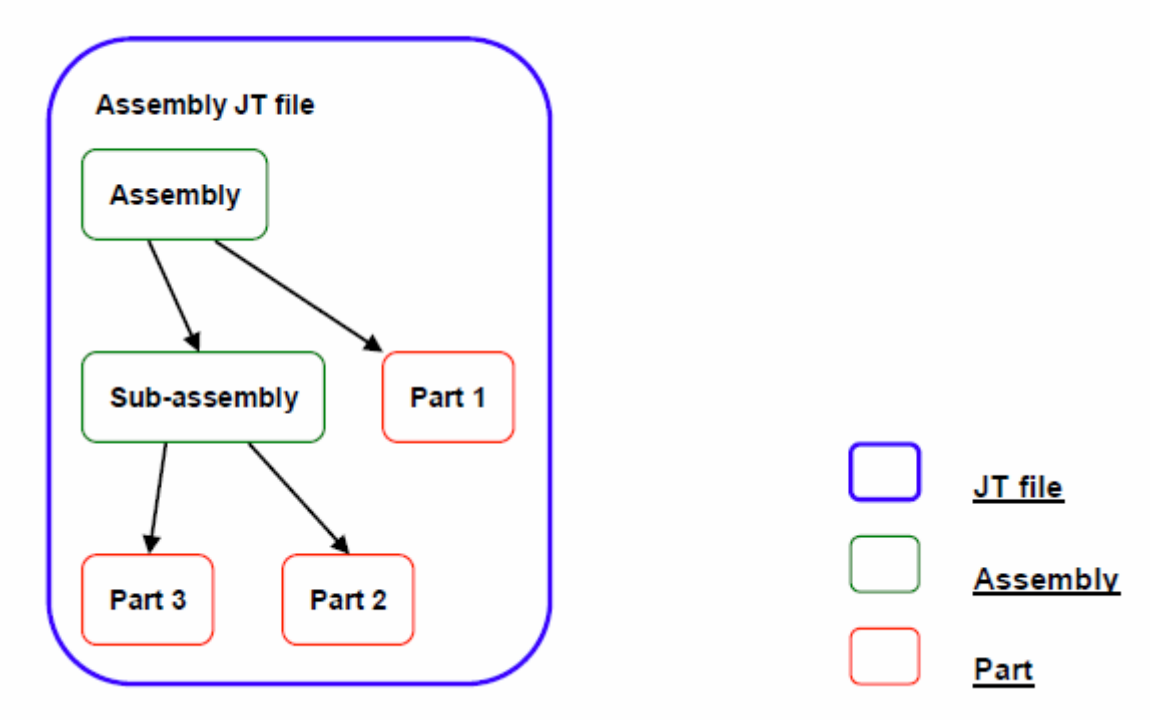


Figure 4: JT only - Monolithic product structure [5]

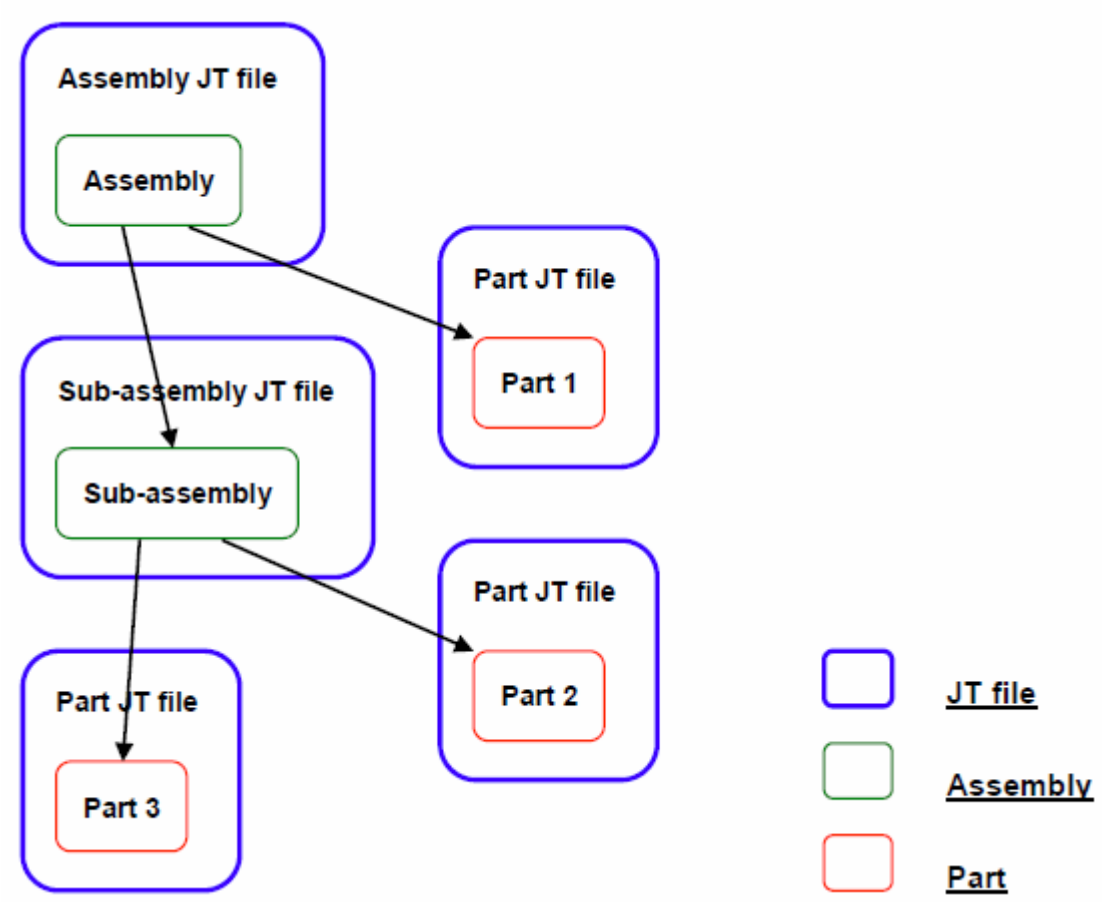


Figure 5: JT only - Fully shattered product structure [5]

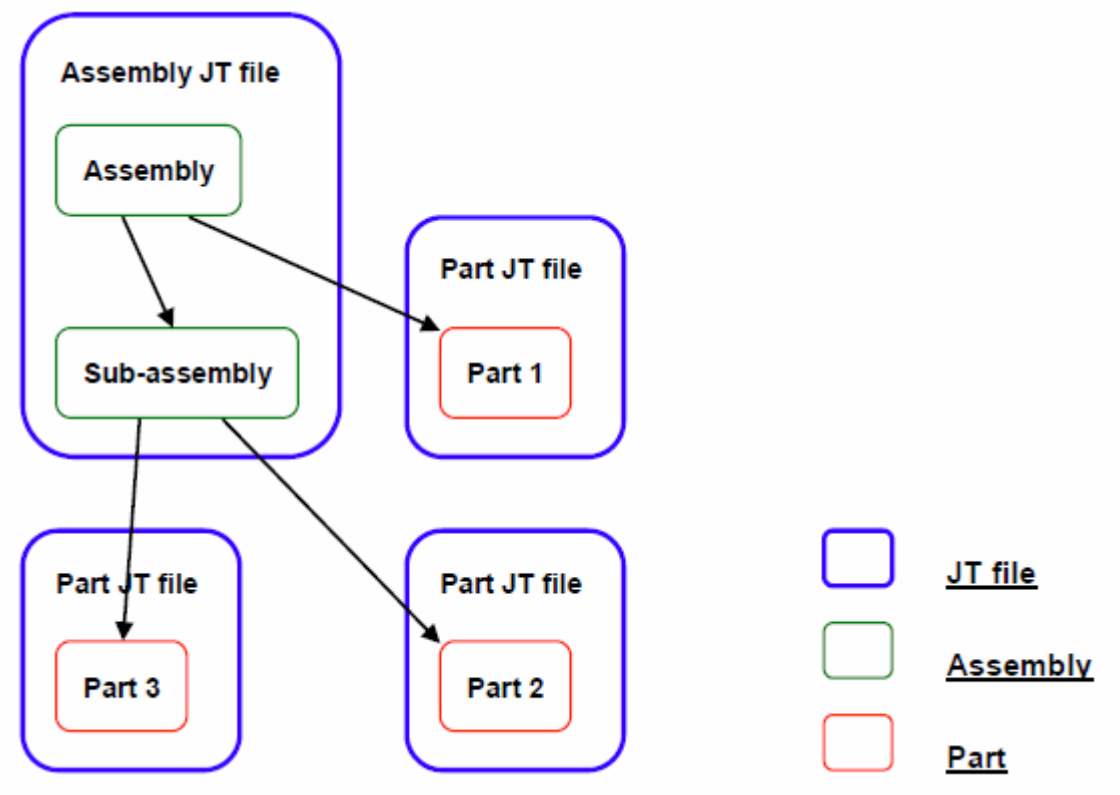


Figure 6: JT only - Per part product structure [5]

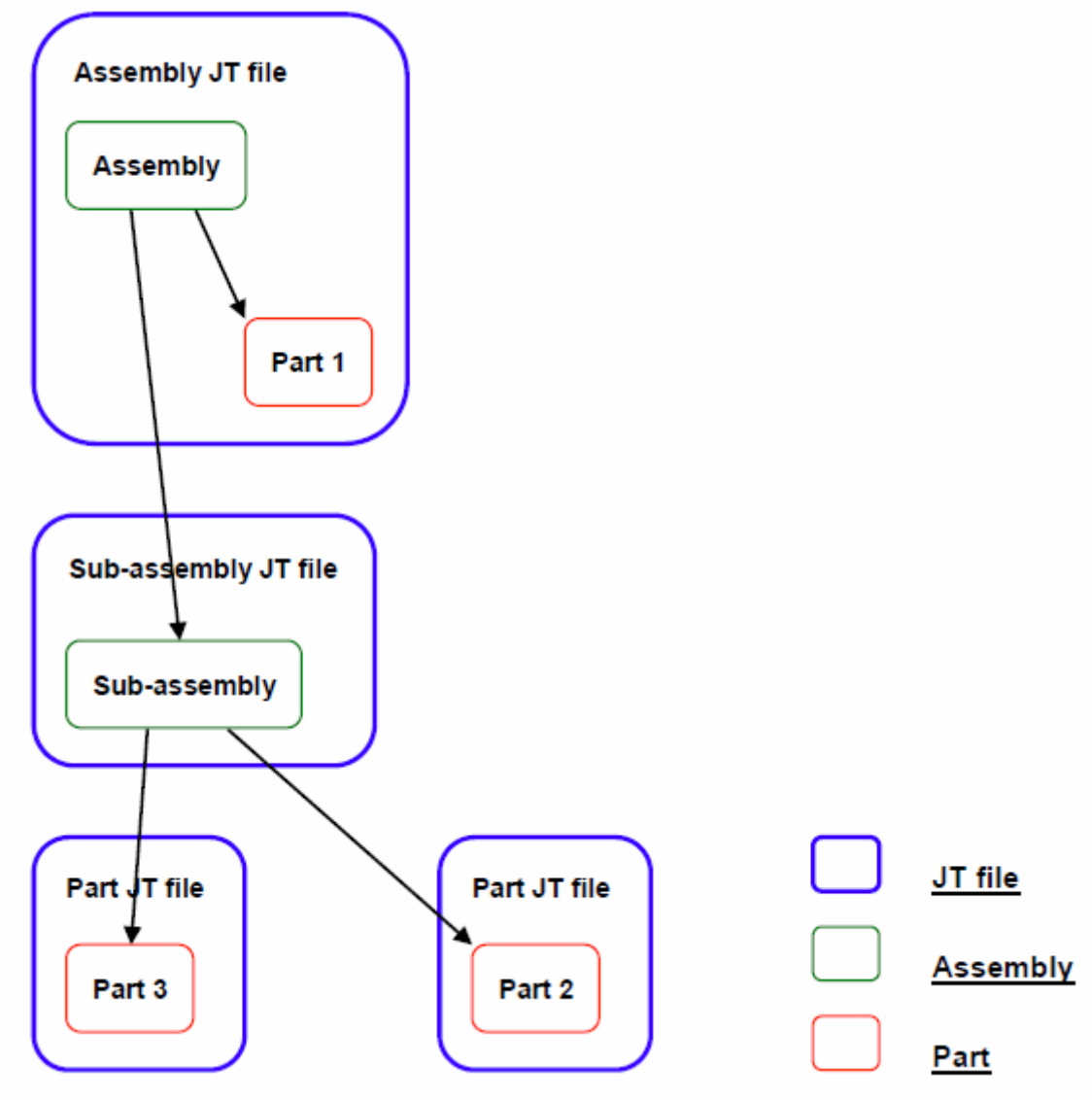


Figure 7: JT only - Single level custom product structure [5]

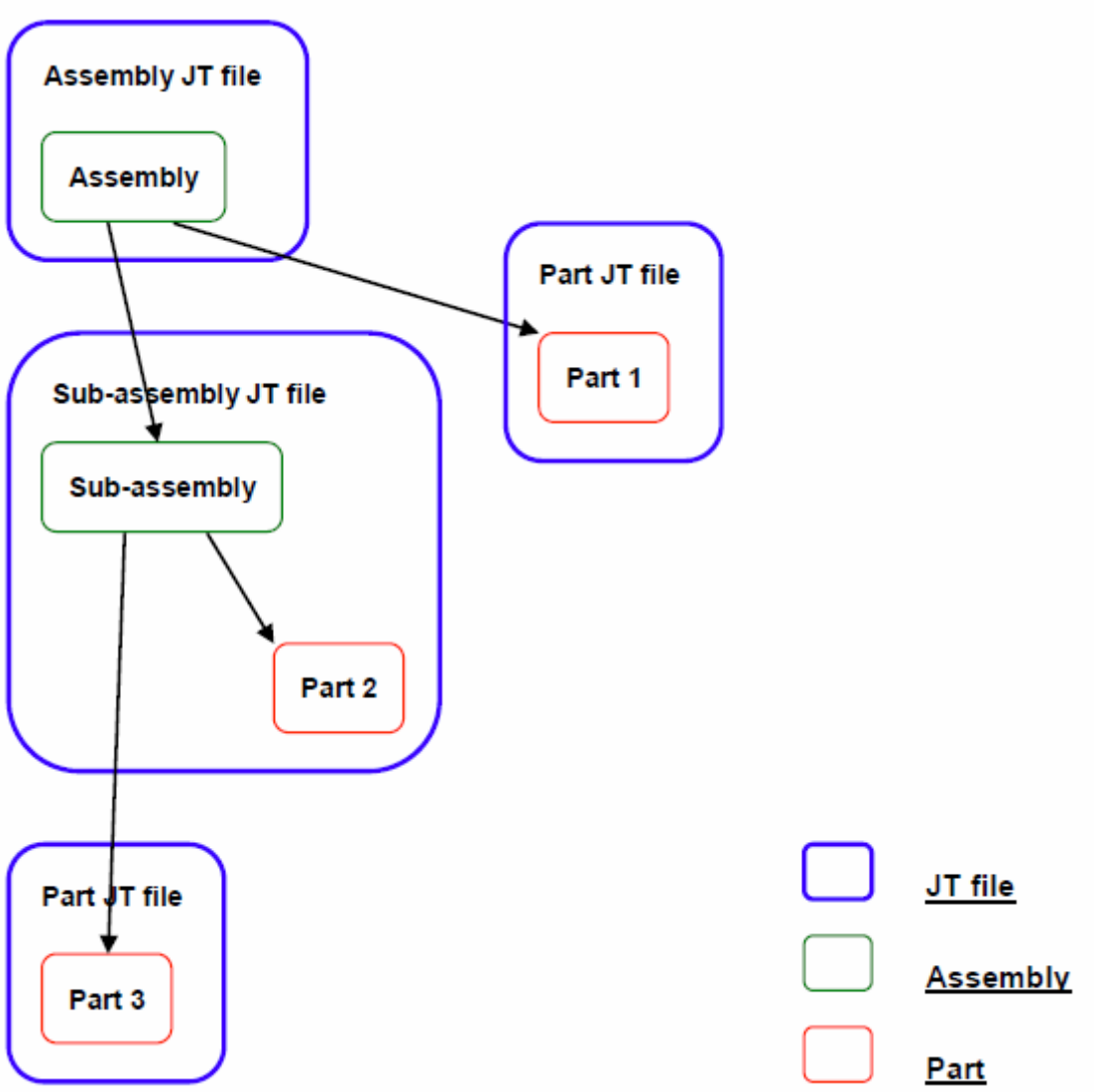


Figure 8: JT only - 2 level custom product structure [5]

The recommended practices for AP242 provide examples as well.

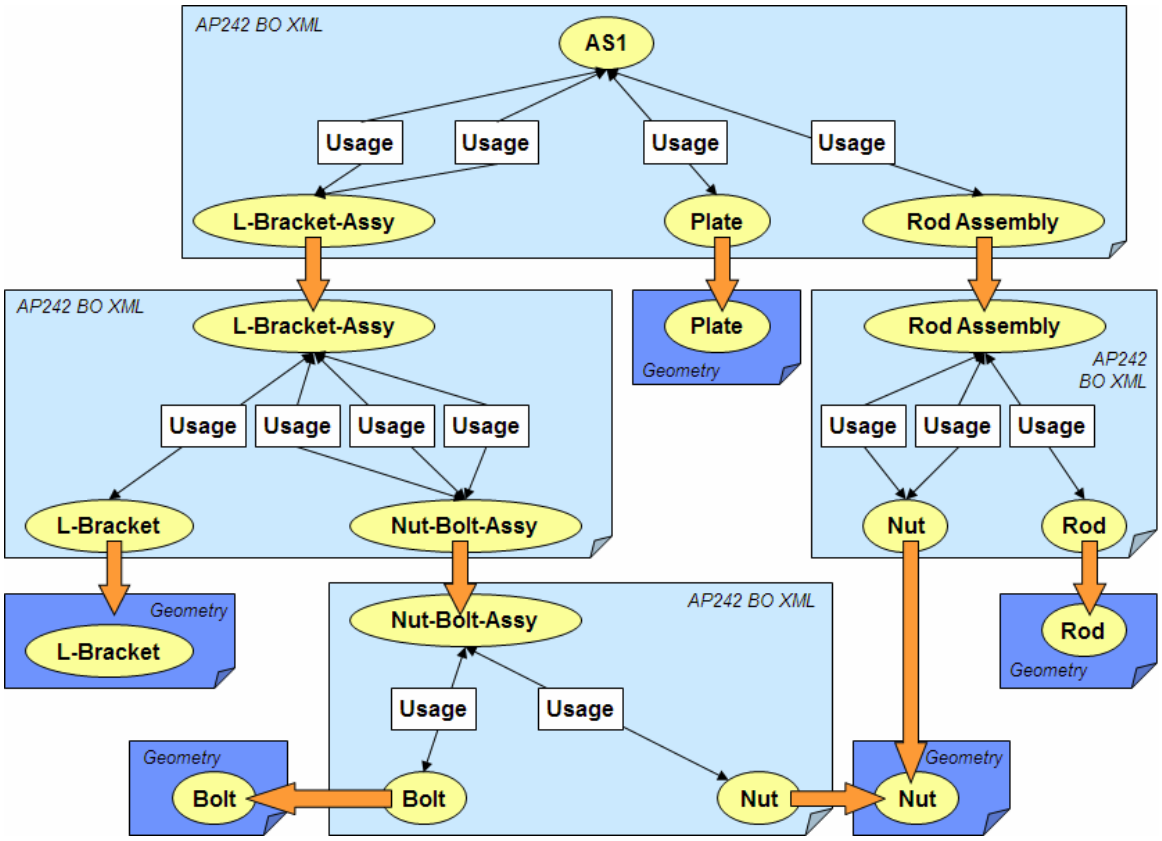


Figure 9: STEP AP242 XML + JT - Nested Structure [6]

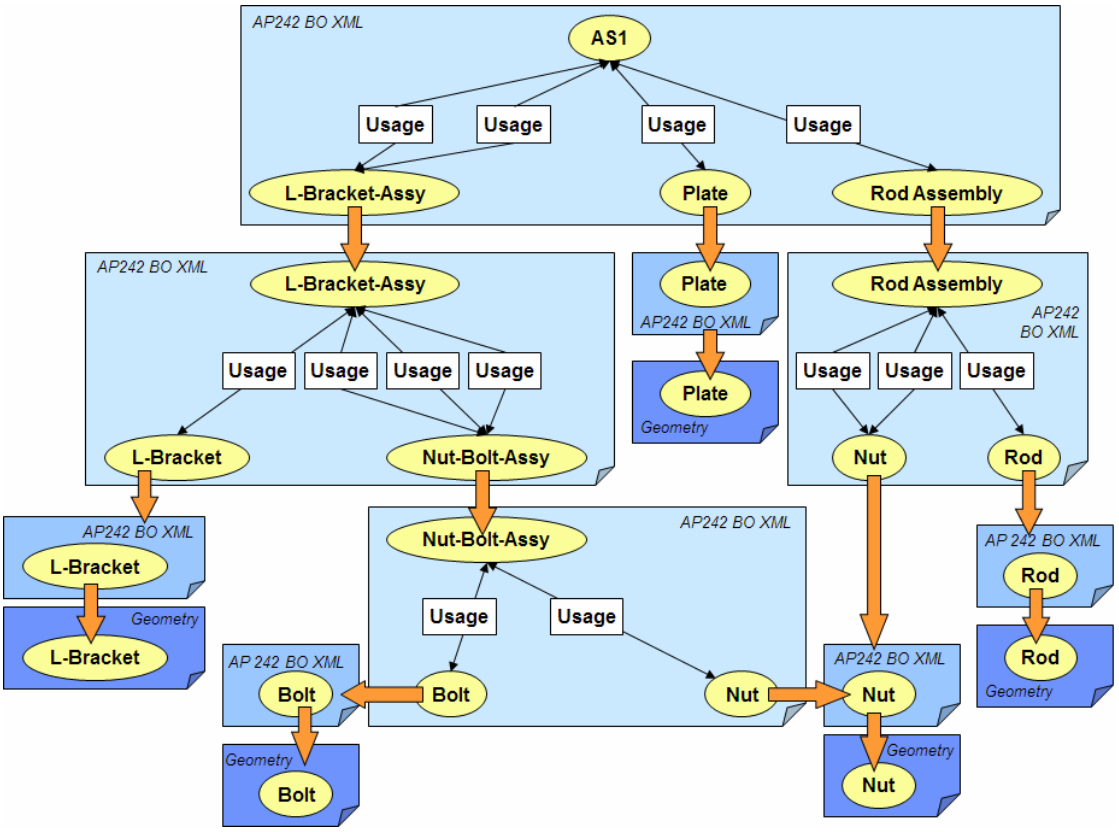


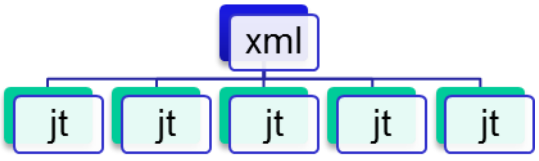
Figure 10: STEP AP242 XML + JT - Nested Structure with additional part-level XML files [6]

8.3.3 JT File Structure Recommendation

The file structures presented in Chapter 8.3.2 are technically possible and do not violate any JT specifications. However, the Content Harmonization Guideline does not recommend to use all technically valid file structures. Therefore, the following list presents recommendations for file structures with regard to JT Use Cases:

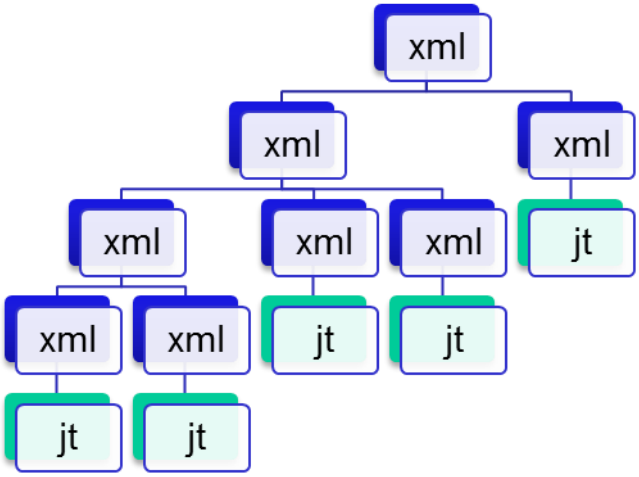
- **Per Part (STEP AP242 XML + JT)**
  - One XML file containing all structure data
  - One JT file for each part node

  Exemplary use cases: Supplier Integration (OEM to Supplier), Supplier Integration (Supplier to OEM)



- **Fully Shattered (STEP AP242 XML + JT)**
  - One XML file for each assembly node
  - One JT file for each part node

  Exemplary use cases: 3D-Measurement and –Analysis, Bidding / Inquiry, High End Visualization, Installation Feasibility, Packaging, Viewing



- **Monolithic (JT only)**
  - One JT file containing all data

  Exemplary use cases: 3D-Measurement and –Analysis, Digital Factory Building Planning, Drawingless Manufacturing, Viewing

8.4 VDA Recommendation 4961/3 and SE-Checklist

The VDA Recommendation 4961/3 supports the CA-organization and data logistics during the cooperation in Simultaneous-Engineering-projects (development partnership). It classifies the models of cooperation and explains the utilization of the SE-Checklist. [8]

Organizational note:

Inside the VDA Recommendation 4961/3 and SE-Checklist there is no detailed information regarding the actual usage of JT. Instead there will be a reference to the JT Content Harmonization Proposal. This way further development of both documents is feasible without special coordination.

Content Harmonization of JT and future versions of this proposal will be done with respect to VDA Recommendation 4961/3 and SE-Checklist.

8.4.1 Objectives of the recommendation:

- Support for the preparation and execution of SE projects
- Consideration of the know-how of all partners and divisions
- Common basis of information
- Cost reduction
- Assistance for partners
- Transparency for the project

8.4.2 Application of the SE-Checklist

The application of the SE-Checklist and related templates is done with respect to the given models of cooperation. The templates support the process of coordination for data logistics in SE-projects.

The SE-Checklist and related templates address the following topics:

- Specific communication and CA infrastructure (CA = Computer Aided) of the project partners
- Definition of data exchange formats for technical documents (data exchange content, - quality, - procedure)
- Process oriented definitions
- Project management, engineering change management
- Definition of target dates, times, costs and responsibilities
- Legal aspects

8.4.3 Occurrence of JT in the SE-Checklist

JT is included in the SE-Checklist at several points. Basically it is listed as an option to choose from along with a number of other formats.

The following sub chapters describe the occurrences of JT in the SE-Checklist. Additionally for each occurrence an example proposal is made how to include JT properly with respect to the Content Harmonization Proposal in general.

8.4.3.1 B. Definition of data exchange formats for technical documents (data exchange content, - quality, - procedure)

TOPIC:

8. Definition of data exchange formats

a) CAD-native
b) Neutral (STEP, VDAFS, …)
c) Visualization formats (JT, …)
d) Raster / postscript formats (TIFF, …)

COMMENT:

Workload optimized definition of data exchange formats; selection of relevant exchange formats depends on use cases and usage of the data.

VDA RECOMMENDATIONS:

VDA-4950, VDA-4966

JT Content Harmonization Proposal:

If JT is chosen as data exchange format, the content of the JT files should be defined prior to the first data exchange. This includes number of LODs, PMI, attributes, B-Rep, etc.

A reliable way to guarantee proper JT content is the agreement on a predefined JT config file with all options and settings made as required. The example config file in chapter 8.2 can be used as template.

The following topics should be discussed and agreed on when choosing JT:

- Which JT version should be used?
- Do you need PMI?
- Do you need exact geometry (B-Reps)?
- How many LODs are required?
- Which quality is required for each LOD?
- Which languages are allowed in the JT for attributes and PMI?

8.4.3.2 C. Process oriented definitions

TOPIC:

13. Regulation of standard documentation during the project, declaring cost break down for additional documentation

COMMENT:

Definition of documentation to be delivered for each project phase (e.g. JT, native, TIF, drawing), cost break down for additional deliverables

JT Content Harmonization Proposal:

If JT is chosen as data exchange format, the content of the JT files should be defined prior to the first data exchange. This includes number of LODs, PMI, attributes, B-Rep, etc.

A reliable way to guarantee proper JT content is the agreement on a predefined JT config file with all options and settings made as required. The example config file in chapter 8.2 can be used as template.

The following topics should be discussed and agreed on when choosing JT:

- Which JT version should be used?
- Do you need PMI?
- Do you need exact geometry (B-Reps)?
- How many LODs are required?
- Which quality is required for each LOD?
- Which languages are allowed in the JT for attributes and PMI?

8.4.3.3 Long-term archiving of CAD documents

JT is in the list of formats to store 3D models. The following options have to be made clear for all formats in the list:

Used format: Yes / no / since when / system, version, comment

Commitment: Only for engineering and development / legally binding

Period of time: Retention period

JT Content Harmonization Proposal:

If JT is chosen as data exchange format, the content of the JT files should be defined prior to the first data exchange. This includes number of LODs, PMI, attributes, B-Rep, etc.

A reliable way to guarantee proper JT content is the agreement on a predefined JT config file with all options and settings made as required. The example config file in chapter 10.2 can be used as template.

The following topics should be discussed and agreed on when choosing JT:

- Which JT version should be used?
- Do you need PMI?
- Do you need exact geometry (B-Reps)?
- How many LODs are required?
- Which quality is required for each LOD?
- Which languages are allowed in the JT for attributes and PMI?

8.4.4 Models of cooperation and JT use cases

One important aspect of the VDA Recommendation 4961/3 and the SE-Checklist is the selection of an appropriate model of cooperation. Depending on the chosen model there are JT use cases with overlapping content and process components. Though a one by one mapping of JT use cases to models of cooperation is not possible due to the different focus of interest, one can still find parallels and benefit from each other by choosing similar JT use cases to the models of cooperation.

Potential JT use cases are:

- Archiving
- Bidding / Inquiry
- Drawingless manufacturing

- Hybrid Design in Context
- Non-hybrid Design in Context
- Supplier Integration (OEM to Supplier)
- Supplier Integration (Supplier to OEM)
- Viewing

The process descriptions and procedures of the use cases may be useful for the coordination of details in the models of cooperation.