

# **Simulation Data Management Annex C: Core data management functionality**

Integration of Simulation and Computation in a PDM-Environment (SimPDM)

Version 2.0, November 2008

Status: Released (22)



## Abstract

This document is Annex C of ProSTEP iViP Recommendation PSI 4. This annex contains the technical description of SimPDM core data management functionality.

The recommendation is intended to standardize the integration of simulation and computation data in a product data management environment (PDM environment) in the form of simulation data management (SDM) and to define the communication processes between a simulation data management system (SDM system) and CAE systems as well as between an SDM system and other data management systems within the PDM environment.

With this purpose in mind, the recommendation defines use cases and functional requirements for simulation data management and a meta data model to describe simulation and computation data. The meta data model is modularly designed to allow a custom-tailored solution for simulation data management. Furthermore this recommendation describes necessary functionality and its clustering towards use case. Functionality and use cases are modular, too. So it is possible to customize SimPDM to the specific requirements of an enterprise.

The recommendation does not define an interchange data format between different CAE applications.

The recommendation consists of several documents:

- The main recommendation document
- Annex A: Glossary (included in the main document)
- Annex B: Business process diagrams (simulation data management for MBS, FEM and CFD)
- Annex C: Core data management functionality (the document in hand)
- Annex D: System communication functionality
- Annex E: Data model

The Annexes B, C, D, E are available for download as separated documents on the websites of ProSTEP iViP e.V ([www.prostep.org](http://www.prostep.org)) and VDA ([www.vda.de](http://www.vda.de)).

## Disclaimer

ProSTEP iViP Recommendations (PSI Recommendations) are recommendations that are available for general use. Anyone using these recommendations is responsible for ensuring that they are used correctly.

This PSI Recommendation gives due consideration to the prevailing state-of-the-art at the time of publication. Anyone using PSI Recommendations 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 Recommendation assume no liability whatsoever.

We request that anyone encountering an error or the possibility of an incorrect interpretation when using the PSI Recommendation should contact the ProSTEP iViP Association ([psi-issues@prostep.com](mailto:psi-issues@prostep.com)) immediately so that any errors can be rectified.

## Copyright

- I. All rights to this PSI Recommendation, in particular the copyright rights of use, and sale such as the right to duplicate, distribute or publish the recommendation, remain exclusively with the ProSTEP iViP Association and its members.
- II. This PSI Recommendation may be duplicated and distributed unchanged, for instance for use in the context of creating software or services.
- III. It is not permitted to change or edit this PSI Recommendation.
- IV. A suitable notice indicating the copyright owner and the restrictions on use must always appear.

This recommendation is also published by the VDA, with the same title and version.

This recommendation has been developed and is supported by the VDA and the ProSTEP iViP Association.

## Acknowledgment

Our thanks go to all the companies and their staff who were actively involved in drafting this recommendation and for the many constructive suggestions received. The following companies and research institutes were involved:

Altair Engineering GmbH, Audi AG, BMW AG, CADFEM Gesellschaft für computerunterstützte Konstruktion und Berechnung mbH, Daimler AG, Das Virtuelle Fahrzeug Forschungsgesellschaft mbH, DiK TU Darmstadt, Dr. Ing. h.c. F. Porsche AG, em engineering methods AG, IBM Deutschland GmbH, IMI Universität Karlsruhe, MAN Turbo AG, MDTVision GmbH, MSC.Software GmbH, PTC GmbH, PDTec AG, PROSTEP AG, Schaeffler KG, Siemens PLM Software, T-Systems Enterprise Services GmbH, Volkswagen AG.

# Contents

## Table of Contents

1 Introduction .....	1
1.1 Scope and Purpose .....	1
1.2 Notation.....	1
1.3 Additional Remarks.....	2
2 Functional description of SimPDM use cases .....	2
2.1 Functionality of use case Administration Management .....	2
2.2 Functionality of use case Analysis Classification Management .....	5
2.3 Functionality of use case Analysis Definition Management.....	6
2.4 Functionality of use case Document Management.....	10
2.5 Functionality of use case Load Case Definition Management .....	15
2.6 Functionality of use case Model Assembly Management.....	20
2.7 Functionality of use case Model Configuration Management.....	21
2.8 Functionality of use case Model Definition Management .....	27
2.9 Functionality of use case Output Specification Management.....	31
2.10 Functionality of use case Parameter Association Management.....	32
2.11 Functionality of use case PDM Information Derivation Management.....	35
2.12 Functionality of use case Post-processing Management .....	41
2.13 Functionality of use case Property Definition Management .....	46
2.14 Functionality of use case Setting Definition Management.....	53
2.15 Functionality of use case Topology Element Definition Management.....	56
2.16 Functionality of use case Topology Structure Definition Management .....	58

# 1 Introduction

## 1.1 Scope and Purpose

The use cases defined by the recommendation are described by their general purpose within the context of simulation data management with regards to their value progress. The describes the atomic data management system functions which are needed to perform the use case, for instance, 'Create new model' as an atomic function of the model definition management. Each function itself is described by the start state and precondition as well as the end state. Each described function requires operations on data objects or the creation of new data objects. These operations are listed for each function and directly refer to the SimPDM data model. The operations are named by a formal code. Last but not least additional notes and remarks are provided, where applicable.

This annex contains the technical description of SimPDM core data management functionality.

## 1.2 Notation

This section gives an overview of the used notation for the SimPDM core data management functionality. It is also valid for Annex D defining SimPDM system communication functionality.

### 1.2.1 Creating new instances of entities of database entities

The creation of new instances of entities is described by the following formal code

- Create.<Package name>.<entity name>

Examples are

- Create.BASE.model\_version
- Create.TOPO.model\_element
- Create.PROP.property\_set

### 1.2.2 Deleting existing instances of database entities

The deletion of new instances of entities is described by the following formal code.

- Delete.<Package name>.<entity name>

Examples are

- Delete.SETT.general\_setting
- Delete.BASE.administration
- Delete.PROP.general\_property

### 1.2.3 Creating new relationships between database objects

The Creation of new relationships between instances of entities is described by the following formal code.

- Connect.<Package name>.<relationship name>

Examples are

- Connect.BASE.result\_first\_order
- Connect.CONF.solution\_model
- Connect.PROP.property\_document

### 1.2.4 Deleting existing relationships between database objects

The deletion of relationships between instances of entities is described by the following formal code.

- Disconnect.<Package name>.<relationship name>

Examples are

- Disconnect.LOAD.applied\_load\_model
- Disconnect.BASE.model\_parameter
- Disconnect.CAD.used\_shape

### 1.2.5 Change of attributes

The setting or changing of attribute values is described by the following formal code

- Set.<package name>.<class name>.<attribute name>

Examples are

- Set.BASE.parameter.name
- Set.BASE.model\_version.description
- Set.PROP.general\_property.property\_specification

### 1.3 Additional Remarks

The document does not claim the completeness of meta data model operations. Especially the change operations for attribute values are only described in a few functions where the attribute value is significant.

The order of the required operations are listed in this document does not imply a mandatory order for the operations to be performed by a data management system.

Generally the meta data model is the fundamental base definition for all described functionality. The required operations directly refer to the meta data model.

## 2 Functional description of SimPDM use cases

### 2.1 Functionality of use case Administration Management

#### 2.1.1 Create new administrative data

##### 2.1.1.1 Start state and preconditions

An instance of one of the following classes, able to carry administrative data, does exist.

- BASE.analysis\_version
- BASE.analysis
- BASE.computation\_output
- BASE.document
- BASE.key\_result
- BASE.model
- BASE.model\_version
- BASE.report
- BASE.template
- CAD.cadpdm\_information
- CONF.configuration
- CONF.product
- CONF.product\_component
- CONF.solution

- LOAD.load\_definition
- PROP.property\_set
- PROP.property\_set\_version
- SETT.setting

### 2.1.1.2 Required operations

- Create.BASE.administration

And, depending on the class instance for which the new instance BASE.administration is to be defined, one of the relationships

- Connect.BASE.analysis\_version\_administration
- Connect.BASE.analysis\_administration
- Connect.BASE.computation\_output\_administration
- Connect.BASE.document\_administration
- Connect.BASE.key\_result\_administration
- Connect.BASE.model\_administration
- Connect.BASE.model\_version\_administration
- Connect.BASE.report\_administration
- Connect.BASE.template\_administration
- Connect.CAD.cad\_or\_pdm\_information\_administration
- Connect.CONF.configuration\_administration
- Connect.CONF.product\_administration
- Connect.CONF.product\_component\_administration
- Connect.CONF.solution\_administration
- Connect.LOAD.load\_definition\_administration
- Connect.PROP.property\_set\_administration
- Connect.PROP.property\_set\_version\_administration
- Connect.SETT.setting\_administration

### 2.1.1.3 End state and post condition

Administrative data is exclusively assigned to an instance of one of the classes

- BASE.analysis\_version
- BASE.analysis
- BASE.computation\_output
- BASE.document
- BASE.key\_result
- BASE.model
- BASE.model\_version
- BASE.report
- BASE.template
- CAD.cadpdm\_information
- CONF.configuration

- CONF.product
- CONF.product\_component
- CONF.solution
- LOAD.load\_definition
- PROP.property\_set
- PROP.property\_set\_version
- SETT.setting

#### **2.1.1.4 Notes and remarks**

No further notes and remarks available

### **2.1.2 Delete existing administrative data**

#### **2.1.2.1 Start state and preconditions**

An instance of the class BASE.administration to be deleted does exist.

#### **2.1.2.2 Required operations**

- Delete.BASE.administration

And, depending on the class instance to which BASE.administration is related, one of

- Disconnect.BASE.analysis\_version\_administration
- Disconnect.BASE.analysis\_administration
- Disconnect.BASE.computation\_output\_administration
- Disconnect.BASE.document\_administration
- Disconnect.BASE.key\_result\_administration
- Disconnect.BASE.model\_administration
- Disconnect.BASE.model\_version\_administration
- Disconnect.BASE.report\_administration
- Disconnect.BASE.template\_administration
- Disconnect.CAD.cadpdm\_information\_administration
- Disconnect.CONF.configuration\_administration
- Disconnect.CONF.product\_administration
- Disconnect.CONF.product\_component\_administration
- Disconnect.CONF.solution\_administration
- Disconnect.LOAD.load\_definition\_administration
- Disconnect.PROP.property\_set\_administration
- Disconnect.PROP.property\_set\_version\_administration
- Disconnect.SETT.setting\_administration

#### **2.1.2.3 End state and post condition**

The administrative data is no longer available.

#### **2.1.2.4 Notes and remarks**

The deletion of an instance of the class `BASE.administration` at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.2 Functionality of use case Analysis Classification Management**

See SimPDM recommendation for a general description of the use case Analysis Classification Management.

### **2.2.1 Create new analysis type**

#### **2.2.1.1 Start state and preconditions**

No specific precondition to be defined.

#### **2.2.1.2 Required operations**

- `Create.BASE.analysis_type`

#### **2.2.1.3 End state and post condition**

A new instance of the class `BASE.analysis_type` is available to support the analysis classification management.

#### **2.2.1.4 Notes and remarks**

Actually, the creation of a new instance of the class `BASE.analysis_type` is not a daily business, but rather an administrative task.

### **2.2.2 Delete existing analysis type**

#### **2.2.2.1 Start state and preconditions**

An instance of the class `BASE.analysis_type` to be deleted does exist and is in a state allowing it to be deleted, i.e., it is not associated with any instance of the class `BASE.analysis` and is not associated with another instance of the class `BASE.analysis_type` with role of a next higher analysis type the within an analysis type hierarchy.

#### **2.2.2.2 Required operations**

- `Delete.BASE.analysis_type`

#### **2.2.2.3 End state and post condition**

The deleted instance of the class `BASE.analysis_type` is no longer available within the data management system.

#### **2.2.2.4 Notes and remarks**

Actually, the deletion of an instance of the class `BASE.analysis_type` is not a daily business, but rather an administrative task.

The deletion of an instance of the class `BASE.analysis_type` at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.2.3 Create new analysis type hierarchy**

#### **2.2.3.1 Start state and preconditions**

At least two instances of the class `BASE.analysis` exist, which have not yet an analysis type hierarchy relationship with each other.

### 2.2.3.2 Required operations

- Connect.BASE.analysis\_type\_hierarchy

### 2.2.3.3 End state and post condition

Two instances of the class BASE.analysis have a hierarchal relationship with each other, where one instance of the class has the role of a sub type and the other instance the role of a next higher analysis type.

### 2.2.3.4 Notes and remarks

Actually, the creation of an analysis type hierarchy is not a daily business, but rather an administrative task.

If the instance of the class BASE.analysis, supposed to be the subtype, already has the role of a subtype in another analysis type hierarchy relationship, the former relationship is implicitly be deleted. An instance of the class BASE.analysis\_type can only be involved in one analysis type hierarchy relationship with the role of a subtype.

## 2.2.4 Delete existing analysis type hierarchy

### 2.2.4.1 Start state and preconditions

An instance of the relationship BASE.analysis\_type\_hierarchy to be deleted does exist and is in a state allowing it to be deleted, i.e., the instance of the class BASE.analysis\_type with the role of the subtype is not associated with an instance of the class BASE.analysis.

### 2.2.4.2 Required operation

- Disconnect.BASE.analysis\_type\_hierarchy

### 2.2.4.3 End state and post condition

The analysis type hierarchy relationship is no longer available within the data management system. The two formerly connected instances of the class BASE.analysis\_type are still available.

### 2.2.4.4 Notes and remarks

Actually, the deletion of an analysis type hierarchy is not a daily business, but rather an administrative task.

The deletion of an analysis type hierarchy at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.3 Functionality of use case Analysis Definition Management

See SimPDM recommendation for a general description of the use case Analysis Definition Management.

### 2.3.1 Create new analysis

#### 2.3.1.1 Start state and preconditions

No start state and precondition to be defined.

#### 2.3.1.2 Required operations

- Create.BASE.analysis
- Create.BASE.analysis\_version
- Connect.BASE.analysis\_version

And optionally

- Connect.BASE.analysis\_classification

### 2.3.1.3 End state and post condition

A new instance of the class `BASE.analysis` and a new initial instance of the class `BASE.analysis_version` for the new instance of the class `BASE.analysis` are available within the data management system.

### 2.3.1.4 Notes and remarks

Instances of the classes `BASE.analysis` and `BASE.analysis_version` are only administrative data management nodes and only have some identifying information and a description. All detailed information for an analysis / simulation are assigned to the instance of the class `BASE.analysis` by separate class instances of different types.

Optionally the use cases Administration Management and Document Management might be applied for the new instances of the classes `BASE.analysis` and `BASE.analysis_version`.

## 2.3.2 Delete existing analysis

### 2.3.2.1 Start state and preconditions

The instance of the class `BASE.analysis` to be deleted has not more than one associated instances class `BASE.analysis_version`, i.e. it has only the initial instance of the class `BASE.analysis_version`. This instance of the class `BASE.analysis_version` has no relationships with other instances of the class `BASE.analysis_version`. This instance of the class `BASE.analysis_version` has no associated information like models, output specification, results; it is not part of one of the relationships `BASE.output_specification`, `BASE.analysis_specific_result` or `BASE.analysis_version_model_version`. The instances of the classes `BASE.analysis` and the `BASE.analysis_version` might have associated instances of the classes `BASE.administration` and `BASE.document`.

### 2.3.2.2 Required operations

- `Delete.BASE.analysis_version`
- `Delete.BASE.analysis`
- `Disconnect.BASE.analysis_version`

And if the instance of the class `BASE.analysis` or the initial instance of the class `BASE.analysis_version` has associated administrative data...

- `Disconnect.BASE.analysis_administration`
- `Disconnect.BASE.analysis_version_administration`
- `Delete.BASE.administration`

And if the initial instance of the class `BASE.analysis_version` has an associated document, all required operation for the deletion of documents (see chapter 2.4.2)

### 2.3.2.3 End state and post condition

The deleted instance of the class `BASE.analysis` is no longer available within the data management system.

### 2.3.2.4 Notes and remarks

The deletion of an instance of the class `BASE.analysis` at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.3.3 Create new analysis version

### 2.3.3.1 Start state and preconditions

An instance of the class `BASE.analysis` with an initial instance of the class `BASE.analysis_version` does exist.

### 2.3.3.2 Required operations

- Create.BASE.analysis\_version
- Connect.BASE.analysis\_version
- Connect.BASE.analysis\_version\_relationship

### 2.3.3.3 End state and post condition

A new instance of the class BASE.analysis\_version for an instance of the class BASE.analysis is available. The new instance of the class BASE.analysis\_version is connected with another already existing instance of the class BASE.analysis\_version of the same instance of the class BASE.analysis with the role as a succeeding version within in a predecessor – successor - relationship.

### 2.3.3.4 Notes and remarks

Optionally the use cases Administration Management and Document Management might be applied for BASE.analysis\_version.

## 2.3.4 Delete existing analysis version

### 2.3.4.1 Start state and preconditions

The instance of the class BASE.analysis\_version to be deleted has no relationships with instances of the class BASE.analysis\_version belonging to separate instances of BASE.analysis, no succeeding instances of the class BASE.analysis\_version, and has no associated information like models, output specification, results, i.e., it is not part of one of the relationships BASE.output\_specification, BASE.analysis\_specific\_result or BASE.analysis\_version\_model\_version. The instances of the classes BASE.analysis and the BASE.analysis\_version might have associated instances of the classes BASE.administration and BASE.document.

### 2.3.4.2 Required operations

- Disconnect.BASE.analysis\_version
- Delete.BASE.analysis\_version

And if the model version has an associated administrative data...

- Disconnect.BASE.analysis\_version\_administration
- Delete.BASE.administration

And if the initial instance of the class BASE.analysis\_version has an associated document, all required operation for the deletion of documents (see chapter 2.4.2)

### 2.3.4.3 End state and post condition

The deleted instance of the class BASE.analysis\_version is no longer available within the data management system.

### 2.3.4.4 Notes and remarks

The deletion of an instance of the class BASE.analysis\_version at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.3.5 Define new analysis version dependency

### 2.3.5.1 Start state and preconditions

Two instances of the class BASE.analysis\_version associated with different instances of the class BASE.analysis do exist.

### 2.3.5.2 Required operations

- Connect.BASE.analysis\_version\_relationship

### 2.3.5.3 End state and post condition

Two instances of the class BASE.analysis\_version associated with different instances of BASE.analysis are associated with each other in a general manner to define a dependency.

### 2.3.5.4 Notes and remarks

No further notes and remarks available

## 2.3.6 Delete existing analysis version dependency

### 2.3.6.1 Start state and preconditions

An instance of the relationship BASE.analysis\_version\_relationship between instances of the class BASE.analysis\_version associated with different instances of the class BASE.analysis does exist.

### 2.3.6.2 Required operations

- Delete.BASE.analysis\_version\_relationship

### 2.3.6.3 End state and post condition

The analysis version dependency defined by an instance of the relationship BASE.analysis\_version\_relationship between two instances of the class BASE.analysis\_version associated with different instances of the class BASE.analysis is no longer available within the data management system.

### 2.3.6.4 Notes and remarks

The deletion of the dependency between two instances of the class BASE.analysis\_version has not any influence on the availability and usability of the formally connected instances of the class BASE.analysis\_version.

## 2.3.7 Classify existing analysis

### 2.3.7.1 Start state and preconditions

An instance of the class BASE.analysis and an instance of the class BASE.analysis\_type do exist.

### 2.3.7.2 Required operations

- Connect.BASE.analysis\_classification

### 2.3.7.3 End state and post condition

An instance of the class BASE.analysis\_type is assigned to the instance of the class BASE.analysis

### 2.3.7.4 Notes and remarks

No further notes and remarks available

## 2.3.8 Delete existing analysis classification

### 2.3.8.1 Start state and preconditions

An instance of the relationship BASE.analysis\_classification between an instance of the class BASE.analysis and an instance of the class BASE.analysis\_type do exist.

### 2.3.8.2 Required operations

- Disconnect.BASE.analysis\_classification

### **2.3.8.3 End state and post condition**

The classification relationship between an instance of the class BASE.analysis and an instance of the class BASE.analysis\_type is deleted.

### **2.3.8.4 Notes and remarks**

No further notes and remarks available

## **2.4 Functionality of use case Document Management**

See SimPDM recommendation for a general description of the use case Document Management.

### **2.4.1 Create new document**

#### **2.4.1.1 Start state and preconditions**

No start state or precondition to be defined

#### **2.4.1.2 Required operations**

- Create.BASE.document

#### **2.4.1.3 End state and post condition**

An instance of the class BASE.document is available within the data management system able to carry one or more instances of the class BASE.reference\_to\_file. The new document is not yet associated to simulation data management objects.

#### **2.4.1.4 Notes and remarks**

Optionally the use case Administration Management might be applied for BASE.document.

### **2.4.2 Delete existing document**

#### **2.4.2.1 Start state and preconditions**

An instance of the class BASE.document to be deleted does exist, must not carry any external files and is in a stage allowing it to be deleted, i.e. it is not in use, respectively must not have relationships with more than one instance of one of the following classes.

- BASE.analysis\_version
- BASE.computation\_output
- BASE.key\_result
- BASE.load\_for\_analysis
- BASE.model\_version
- BASE.output\_request
- BASE.report
- BASE.setting\_for\_analysis
- BASE.template
- CAD.cadpdm\_information
- LOAD.load\_definition
- PROP.property
- PROP.property\_constraint
- SETT.setting
- TOPO.model\_element

#### **2.4.2.2 Required operations**

- Delete.BASE.document

And if the document has an associated administration object

- Disconnect.BASE.document\_administration
- Delete.BASE.administration

#### **2.4.2.3 End state and post condition**

The deleted instance of the class BASE.document is no longer available within the data management system.

#### **2.4.2.4 Notes and remarks**

The deletion instances of the class BASE.document at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.4.3 Create new document association**

#### **2.4.3.1 Start state and preconditions**

An instance of the class BASE.document and at least one instance of one of the following classes does exist.

- BASE.analysis\_version
- BASE.computation\_output
- BASE.key\_result
- BASE.load\_for\_analysis
- BASE.model\_version
- BASE.output\_request
- BASE.report
- BASE.setting\_for\_analysis
- BASE.template
- CAD.cadpdm\_information
- LOAD.load\_definition
- PROP.property
- PROP.property\_constraint
- SETT.setting
- TOPO.model\_element

#### **2.4.3.2 Required operations**

Depending on the class instance to which the document is to be associated, one of

- Connect.BASE.analysis\_version\_document
- Connect.BASE.computation\_output\_document
- Connect.BASE.key\_result\_document
- Connect.BASE.load\_for\_analysis\_document
- Connect.BASE.model\_version\_document
- Connect.BASE.output\_request\_document

- Connect.BASE.report\_document
- Connect.BASE.setting\_for\_analysis\_document
- Connect.BASE.template\_document
- Connect.CAD.cad\_or\_pdm\_information\_document
- Connect.LOAD.load\_definition\_document
- Connect.PROP.property\_document
- Connect.PROP.property\_constraint\_document
- Connect.SETT.setting\_document
- Connect.TOPO.model\_element\_document

### 2.4.3.3 End state and post condition

An instance of the class BASE.document is associated to one instance of one of the following classes.

- BASE.analysis\_version
- BASE.computation\_output
- BASE.key\_result
- BASE.load\_for\_analysis
- BASE.model\_version
- BASE.output\_request
- BASE.report
- BASE.setting\_for\_analysis
- BASE.template
- CAD.cadpdm\_information
- LOAD.load\_definition
- PROP.property
- PROP.property\_constraint
- SETT.setting
- TOPO.model\_element

### 2.4.3.4 Notes and remarks

An instance of BASE.document can only have one exclusive relationship to a specific instance of one of the classes listed above. Nevertheless, an instance of the class BASE.document might have several relationships to instances of different classes or different instances of one and the same class.

## 2.4.4 Delete existing document association

### 2.4.4.1 Start state and preconditions

A relationship between an instance of the class BASE.document and one instance of one of the following classes does exist,

- BASE.analysis\_version
- BASE.computation\_output
- BASE.key\_result
- BASE.load\_for\_analysis
- BASE.model\_version

- BASE.output\_request
- BASE.report
- BASE.setting\_for\_analysis
- BASE.template
- CAD.cadpdm\_information
- LOAD.load\_definition
- PROP.property
- PROP.property\_constraint
- SETT.setting
- TOPO.model\_element

#### **2.4.4.2 Required operations**

Depending on the class instance to which BASE.documentation is related, one of

- Disconnect.BASE.analysis\_version\_document
- BASE.computation\_output
- BASE.key\_result
- BASE.load\_for\_analysis
- Disconnect.BASE.model\_version\_document
- Disconnect.BASE.output\_request\_document
- BASE.report
- BASE.setting\_for\_analysis
- Disconnect.BASE.template\_document
- Disconnect.CAD.cad\_or\_pdm\_information\_document
- Disconnect.LOAD.load\_definition\_document
- Disconnect.PROP.property\_document
- Disconnect.PROP.property\_constraint\_document
- Disconnect.SETT.setting\_document
- Disconnect.TOPO.model\_element\_document

#### **2.4.4.3 End state and post condition**

A specific instance of the class BASE.document is no longer associated with one specific instance of one of the following classes.

- BASE.analysis\_version
- BASE.computation\_output
- BASE.key\_result
- BASE.load\_for\_analysis
- BASE.model\_version
- BASE.output\_request
- BASE.report
- BASE.setting\_for\_analysis
- BASE.template

- CAD.cadpdm\_information
- LOAD.load\_definition
- PROP.property
- PROP.property\_constraint
- SETT.setting
- TOPO.model\_element

Nevertheless, the instance of the class BASE.document still exists and may still have associations to other instances of one of the listed classes.

#### **2.4.4.4 Notes and remarks**

The deletion of relationships at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.4.5 Attach external file reference**

#### **2.4.5.1 Start state and preconditions**

An instance of the class BASE.document as an administrative container object (“external file in context”) exists.

#### **2.4.5.2 Required operations**

- Create.BASE.reference\_to\_file
- Connect.BASE.document\_file

#### **2.4.5.3 End state and post condition**

An instance of the class BASE.reference\_to\_file is available within the data management system and is, using an instance of the class BASE.document as the administrative container („external file in context“), associated with an instance of the one of the following classes.

- BASE.analysis\_version
- BASE.computation\_output
- BASE.key\_result
- BASE.load\_for\_analysis
- BASE.model\_version
- BASE.output\_request
- BASE.report
- BASE.setting\_for\_analysis
- BASE.template
- CAD.cadpdm\_information
- LOAD.load\_definition
- PROP.property
- PROP.property\_constraint
- SETT.setting
- TOPO.model\_element

#### **2.4.5.4 Notes and remarks**

An instance of the class `BASE.document` can carry more than one instances of the class `BASE.reference_to_file` but an instance of the class `BASE.reference_to_file` can only be associated to one instance of the class `BASE.document`. Nevertheless, an instance of the class `BASE.document` should not be redundantly associated with more than one instance of the class `BASE.reference_to_file` referring to the same external file. Within a data management system a specific external file can be referred by more than one instance of the class `BASE.reference_to_file` associated to different instances of the class `BASE.document`.

#### **2.4.6 Delete external files reference**

##### **2.4.6.1 Start state and preconditions**

An instance of the class `BASE.reference_to_file` does exist.

##### **2.4.6.2 Required operations**

- `Delete.BASE.reference_to_file`
- `Disconnect.BASE.document_file`

##### **2.4.6.3 End state and post condition**

The instance of the class `BASE.reference_to_file` is no longer available within the data management system.

##### **2.4.6.4 Notes and remarks**

The SimPDM data model does not define the deletion of the external file itself, since deletion of files not only depends on the data management functionalities and rights but also on operation system functionalities and rights.

### **2.5 Functionality of use case Load Case Definition Management**

See SimPDM recommendation for a general description of the use case Load Case Definition Management.

#### **2.5.1 Create new load**

##### **2.5.1.1 Start state and preconditions**

No start state and precondition to be defined.

##### **2.5.1.2 Required operations**

- `Create.LOAD.load_definition` (or one of the subtypes)

##### **2.5.1.3 End state and post condition**

A new `LOAD.load_definition` definition is available within the data management system.

##### **2.5.1.4 Notes and remarks**

No further notes and remarks available.

#### **2.5.2 Delete existing load**

##### **2.5.2.1 Start state and preconditions**

A `LOAD.load_definition` to be deleted does exist and is in a stage allowing change, i.e. it has no further objects like `BASE.document`, etc. assigned. The `LOAD.load_definition` might have an associated `BASE.administration` which is also subject to be deleted then (see chapter 2.1).

##### **2.5.2.2 Required operations**

- `Delete.LOAD.load_definition` (or one of the subtypes)

### **2.5.2.3 End state and post condition**

The LOAD.load\_definition definition is no longer available within the data management system.

### **2.5.2.4 Notes and remarks**

The deletion of LOAD.load\_definition at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.5.3 Assign property to load**

### **2.5.3.1 Start state and preconditions**

A LOAD.load\_definition or one of its subtypes exists and is in a state allowing change. Furthermore a PROP.property\_set\_version does exist.

### **2.5.3.2 Required operations**

- Connect.LOAD.load\_property\_set\_version

### **2.5.3.3 End state and post condition**

A LOAD.load\_definition or one of its subtypes has assigned a PROP.property\_set\_version within the data management system.

### **2.5.3.4 Notes and remarks**

The PROP.property\_set\_version already existed within the data management system. Creation is subject of another use case (see chapter 2.13).

## **2.5.4 Delete property assignment from load**

### **2.5.4.1 Start state and preconditions**

A LOAD.load\_definition or one of its subtypes has assigned a PROP.property\_set\_version and is in a state allowing change.

### **2.5.4.2 Required operations**

- Disconnect.LOAD.load\_property\_set\_version

### **2.5.4.3 End state and post condition**

A LOAD.load\_definition or one of its subtypes has no longer assigned a PROP.property\_set\_version within the data management system.

### **2.5.4.4 Notes and remarks**

The PROP.property\_set\_version still exists within the data management system. Deletion is subject of another use case (see chapter 2.13).

The deletion of the relation LOAD.load\_property\_set\_version at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.5.5 Assign load to a model element**

### **2.5.5.1 Start state and preconditions**

A LOAD.load\_definition or one of its subtypes exists and is in a state allowing change. Furthermore a TOPO.model\_element or one of its subtypes does exist.

### 2.5.5.2 Required operations

- Connect.LOAD.load\_definition\_model\_element

### 2.5.5.3 End state and post condition

A LOAD.load\_definition or one of its subtypes has assigned a TOPO.model\_element or one of its subtypes within the data management system.

### 2.5.5.4 Notes and remarks

The TOPO.model\_element or one of its subtypes already existed within the data management system. Creation is subject of another use case (see chapter 2.15).

## 2.5.6 Delete load assignment from model element

### 2.5.6.1 Start state and preconditions

A LOAD.load\_definition or one of its subtypes has assigned a TOPO.model\_element or one of its subtypes and is in a state allowing change.

### 2.5.6.2 Required operations

- Disconnect.LOAD.load\_definition\_model\_element

### 2.5.6.3 End state and post condition

A LOAD.load\_definition or one of its subtypes has no longer assigned a TOPO.model\_element or one of its subtypes within the data management system.

### 2.5.6.4 Notes and remarks

The TOPO.model\_element or one of its subtypes still exists within the data management system. Deletion is subject of another use case (see chapter 2.15).

The deletion of the relation LOAD.load\_definition\_model\_element at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.5.7 Assign a model as load (e.g. dummy, barrier)

### 2.5.7.1 Start state and preconditions

A LOAD.applied\_load\_case exists and is in a state allowing change. Furthermore a BASE.model\_version does exist.

### 2.5.7.2 Required operations

- Connect.LOAD.applied\_load\_model

### 2.5.7.3 End state and post condition

A LOAD.applied\_load\_case has assigned a BASE.model\_version within the data management system.

### 2.5.7.4 Notes and remarks

No further notes and remarks available.

## 2.5.8 Delete load model assignment from load

### 2.5.8.1 Start state and preconditions

A LOAD.applied\_load\_case has assigned a BASE.model\_version and is in a state allowing change.

### 2.5.8.2 Required operations

- Disconnect.LOAD.applied\_load\_model

### 2.5.8.3 End state and post condition

A LOAD.applied\_load\_case or one of its subtypes has no longer assigned a BASE.model\_version within the data management system.

### 2.5.8.4 Notes and remarks

The LOAD.applied\_load\_case and the BASE.model\_version still exist within the data management system. Deletion is subject of another use case (see chapter 2.8).

The deletion of the relation .LOAD.applied\_load\_model at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.5.9 Assign load to an analysis

### 2.5.9.1 Start state and preconditions

A BASE.analysis\_version does exist and is in a state allowing change. For high granularity load management existence of LOAD.load\_definition or one of its subtypes can be necessary.

### 2.5.9.2 Required operations

- Create.BASE.load\_for\_analysis
- Connect.BASE.load\_for\_analysis\_version

And, in case of high granularity level data management, additionally

- Create.LOAD.applied\_load\_case
- Connect.LOAD.analysis\_applied\_load\_case

And, at least one of ...

- Connect.LOAD.load\_applied\_load\_case
- Connect.LOAD.boundary\_condition\_applied\_load\_case
- Connect.LOAD.script\_applied\_load\_case
- Connect.LOAD.start\_condition\_applied\_load\_case

### 2.5.9.3 End state and post condition

A BASE.load\_for\_analysis is assigned to a BASE.analysis\_version. In case of high level granularity this is further detailed in the assigned LOAD.applied\_load\_case and its assignments.

### 2.5.9.4 Notes and remarks

BASE.load\_for\_analysis can be defined in a document. Document attachment in general is subject of another use case (see chapter 2.4).

## 2.5.10 Delete load assignment from analysis

### 2.5.10.1 Start state and preconditions

A BASE.load\_for\_analysis is assigned to a BASE.analysis\_version which is in a state allowing change. In case of high level granularity the load is further detailed in the assigned LOAD.applied\_load\_case and its assignments which also exist and are in a state allowing change. There is no BASE.document attached to BASE.load\_for\_analysis (see chapter 2.4).

### 2.5.10.2 Required operations

- Delete.BASE.load\_for\_analysis
- Disconnect.BASE.load\_for\_analysis\_version

And, in case of high granularity level data management, additionally

- Delete.LOAD.applied\_load\_case
- Disconnect.LOAD.analysis\_applied\_load\_case

And, depending on assigned loads,

- Disconnect.LOAD.load\_applied\_load\_case
- Disconnect.LOAD.boundary\_condition\_applied\_load\_case
- Disconnect.LOAD.script\_applied\_load\_case
- Disconnect.LOAD.start\_condition\_applied\_load\_case

### 2.5.10.3 End state and post condition

A BASE.load\_for\_analysis is no longer available within the data management system. Also no longer available are the LOAD.applied\_load\_case and the assignments.

### 2.5.10.4 Notes and remarks

The LOAD.load\_definition or one of its subtypes and the BASE.model\_version still exist within the data management system. Deletion is subject of another use case (see chapter 2.8).

The deletion of load assignments at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.5.11 Add another detailed load to an analysis

### 2.5.11.1 Start state and preconditions

A LOAD.applied\_load\_case does exist and is in a state allowing change. A LOAD.load\_definition or one of its subtypes exists.

### 2.5.11.2 Required operations

One of ...

- Connect.LOAD.load\_applied\_load\_case
- Connect.LOAD.boundary\_condition\_applied\_load\_case
- Connect.LOAD.script\_applied\_load\_case
- Connect.LOAD.start\_condition\_applied\_load\_case

### 2.5.11.3 End state and post condition

Another LOAD.load\_definition or one of its subtypes is assigned to a LOAD.applied\_load\_case within the data management system.

### 2.5.11.4 Notes and remarks

No further notes and remarks available.

## 2.5.12 Delete a detailed load assignment

### 2.5.12.1 Start state and preconditions

A LOAD.load\_definitions or one of its subtypes are assigned to a LOAD.applied\_load\_case within the data management system. The LOAD.applied\_load\_case is in a state allowing change.

### 2.5.12.2 Required operations

Depending on assigned loads, one of ...

- Disconnect.LOAD.load\_applied\_load\_case
- Disconnect.LOAD.boundary\_condition\_applied\_load\_case
- Disconnect.LOAD.script\_applied\_load\_case
- Disconnect.LOAD.start\_condition\_applied\_load\_case

### 2.5.12.3 End state and post condition

A LOAD.load\_definition or one of its subtypes is no longer assigned to a LOAD.applied\_load\_case within the data management system.

### 2.5.12.4 Notes and remarks

The LOAD.load\_definition or one of its subtypes and the LOAD.applied\_load\_case still exist within the data management system.

The deletion of load assignments at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.6 Functionality of use case Model Assembly Management

See SimPDM recommendation for a general description of the use case Model Assembly Management.

### 2.6.1 Create new model assembly relationship

#### 2.6.1.1 Start state and preconditions

At least two instances of the class BASE.model\_version of different instances of the class BASE.model exist, which do not yet have a model assembly relationship with each other.

#### 2.6.1.2 Required operations

- Create.BASE.model\_version\_relationship\_with\_transformation
- Create.BASE.transformation
- Connect.BASE.related\_model
- Connect.BASE.relatng\_model
- Connect.BASE.transformation

#### 2.6.1.3 End state and post condition

A new model assembly relationship between two instances of the class BASE.model is available. The sub model is positioned within the frame of the next higher model.

#### 2.6.1.4 Notes and remarks

The instance of the class BASE.transformation does only define the positioning of the submodel frame within the frame of the next higher model. The instance of the class transformation does not define the topological connection between two instances of the class model. More else, SimPDM does not enforce the definition of a model topology or the topological connection on the data management level. Alternatively, the model topology and topological connection between models might also be defined in external model files, which are managed by an instance of the class BASE.document and instances of the class BASE.reference\_to\_file.

The actual coefficients of the transformation are defined by instances of the class BASE.parameter assigned to the instance of the class BASE.transformation. See chapter 2.9 for the functionality of parameter association management.

## 2.6.2 Delete existing model assembly relationship

### 2.6.2.1 Start state and preconditions

An instance of the class `BASE.model_version_relationship_with_transformation` as a definition of a model assembly hierarchy definition to be deleted does exist. The next higher instance of the class `BASE.model_version` is in a state allowing it to be changed, i.e., it is not used by an instance of the class `BASE.analysis_version` which does not allow changes.

### 2.6.2.2 Required operations

- `Disconnect.BASE.transformation`
- `Delete.BASE.transformation`
- `Disconnect.BASE.related_model`
- `Disconnect.BASE.relatng_model`
- `Delete.BASE.model_version_relationship_with_transformation`

### 2.6.2.3 End state and post condition

The relationship between the two instances of the class `BASE.model_version` is entirely deleted, including the instance of the class `BASE.transformation`. The two instances of the class `BASE.model_version` are still available. Deletion of `BASE.parameter` is subject of another use case (see chapter 2.10).

### 2.6.2.4 Notes and remarks

The deletion of a model assembly relationship at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.7 Functionality of use case Model Configuration Management

See SimPDM recommendation for a general description of the use case Model Configuration Management. The use case Model Configuration Management is not a major use case within SimPDM (though it is a very important topic within product development and overall product data management). Therefore this use case includes functionalities for the management of an abstract product structure (chapters 2.7.1 to 2.7.10) as well as for the management and procedure of a configuration itself (chapters 2.7.11 to 2.7.18).

### 2.7.1 Create abstract product

#### 2.7.1.1 Start state and preconditions

No start state and precondition to be defined.

#### 2.7.1.2 Required operations

- `Create.CONF.product`

#### 2.7.1.3 End state and post condition

A new `CONF.product` is available to support the abstract product structure.

#### 2.7.1.4 Notes and remarks

Actually, the creation of a `CONF.product` is not a daily business, but rather an administrative task.

## 2.7.2 Delete abstract product

### 2.7.2.1 Start state and preconditions

A CONF.product does exist and is in a state allowing it to be deleted, i.e. it has no CONF.product\_component assigned. The CONF.product might have an associated BASE.administration which is also subject to be deleted then (see chapter 2.1).

### 2.7.2.2 Required operations

- Delete.CONF.product

### 2.7.2.3 End state and post condition

The CONF.product is no longer available within the data management system.

### 2.7.2.4 Notes and remarks

Actually, the deletion of a CONF.product is not a daily business, but rather an administrative task.

The deletion of CONF.product at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.7.3 Create product component

### 2.7.3.1 Start state and preconditions

A CONF.product does exist and is in a stage allowing changes.

### 2.7.3.2 Required operations

- Create.CONF.product\_component
- Connect.CONF.product\_product\_component

### 2.7.3.3 End state and post condition

A new product component CONF.product\_component is available and connected to the abstract product CONF.product.

### 2.7.3.4 Notes and remarks

Actually, the creation of a CONF.product\_component is not a daily business, but rather an administrative task.

In case of creating a CONF.product\_component hierarchy, the assignment of the CONF.product\_component to a CONF.product is needless.

## 2.7.4 Delete product component

### 2.7.4.1 Start state and preconditions

A CONF.product\_component does exist and is in a state allowing it to be deleted, i.e. it has no CONF.product\_component and/or CONF.solution assigned. The CONF.product\_component might have an associated BASE.administration which is also subject to be deleted then (see chapter 2.1).

### 2.7.4.2 Required operations

- Disconnect.CONF.product\_product\_component
- Delete.CONF.product\_component

### 2.7.4.3 End state and post condition

The CONF.product\_component is no longer available within the data management system.

#### **2.7.4.4 Notes and remarks**

Actually, the deletion of a CONF.product\_component is not a daily business, but rather an administrative task.

The deletion of CONF.product\_component at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.7.5 Create component hierarchy**

#### **2.7.5.1 Start state and preconditions**

Two CONF.product\_component exist and are in a stage allowing changes. One of these CONF.product\_component is not assigned to a CONF.product.

#### **2.7.5.2 Required operations**

- Connect.CONF.product\_component\_hierarchy

#### **2.7.5.3 End state and post condition**

A new product component hierarchy is available within the data management system.

#### **2.7.5.4 Notes and remarks**

Actually, the creation of component hierarchies is not a daily business, but rather an administrative task.

### **2.7.6 Delete component hierarchy**

#### **2.7.6.1 Start state and preconditions**

Two CONF.product\_component which are related exist and are in a state allowing them to be deleted, i.e. the lower CONF.product\_component has no further CONF.product\_component and/or CONF.solution assigned. The lower CONF.product\_component might have associated BASE.administration which is also subject to be deleted then (see chapter 2.1).

#### **2.7.6.2 Required operations**

- Disconnect.CONF.product\_component\_hierarchy
- Delete.CONF.product\_component (Deletion of the lower CONF.product\_component)

#### **2.7.6.3 End state and post condition**

The product component hierarchy is no longer available within the data management system.

#### **2.7.6.4 Notes and remarks**

Actually, the deletion of a component hierarchy is not a daily business, but rather an administrative task.

The deletion of a component hierarchy at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.7.7 Create solution**

#### **2.7.7.1 Start state and preconditions**

A CONF.product\_component does exist and is in a stage allowing changes.

#### **2.7.7.2 Required operations**

- Create.CONF.solution
- Connect.CONF.product\_component\_solution

### **2.7.7.3 End state and post condition**

A new solution CONF.solution is available and connected to the abstract product component CONF.product\_component.

### **2.7.7.4 Notes and remarks**

Actually, the creation of a CONF.solution is not a daily business, but rather an administrative task.

## **2.7.8 Delete solution**

### **2.7.8.1 Start state and preconditions**

A CONF.solution does exist and is in a state allowing it to be deleted, i.e. it has no BASE.model assigned and/or is not used by a CONF.configuration. The CONF.solution might have an associated BASE.administration which is also subject to be deleted then (see chapter 2.1).

### **2.7.8.2 Required operations**

- Disconnect.CONF.product\_component\_solution
- Delete.CONF.product\_solution

### **2.7.8.3 End state and post condition**

The CONF.solution is no longer available within the data management system.

### **2.7.8.4 Notes and remarks**

Actually, the deletion of a CONF.solution is not a daily business, but rather an administrative task.

The deletion of CONF.solution at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.7.9 Assign model to solution**

### **2.7.9.1 Start state and preconditions**

A BASE.model does exist. A CONF.solution does exist and both are in a stage allowing changes.

### **2.7.9.2 Required operations**

- Connect.CONF.solution\_model

### **2.7.9.3 End state and post condition**

A CONF.solution is related to a BASE.model within the data management system.

### **2.7.9.4 Notes and remarks**

No further notes and remarks available.

## **2.7.10 Delete model assignment from solution**

### **2.7.10.1 Start state and preconditions**

A CONF.solution and a BASE.model which are related exist and are in a state allowing them to be disconnected.

### **2.7.10.2 Required operations**

- Disconnect.CONF.solution\_model

### **2.7.10.3 End state and post condition**

A BASE.model is no longer included into a CONF.solution. Within the data management system both objects still exist on their own.

### **2.7.10.4 Notes and remarks**

The deletion of the relation CONF.solution\_model at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.7.11 Create configuration within SimPDM**

### **2.7.11.1 Start state and preconditions**

One or more CONF.solution exist.

### **2.7.11.2 Required operations**

- Create.CONF.configuration
- Connect.CONF.configuration\_solution

And, if the configuration is hierarchic, additionally

- Connect.CONF.configuration\_hierarchy

### **2.7.11.3 End state and post condition**

A CONF.configuration with assigned CONF.solution is available within the data management system.

### **2.7.11.4 Notes and remarks**

No further notes and remarks available.

## **2.7.12 Assign configuration to BOM**

### **2.7.12.1 Start state and preconditions**

A CAD.bill\_of\_material exists. A CONF.configuration exists and is in a state allowing change.

### **2.7.12.2 Required operations**

- Connect.CONF.configuration\_bom

### **2.7.12.3 End state and post condition**

A CONF.configuration is assigned to a CAD.bill\_of\_material.

### **2.7.12.4 Notes and remarks**

No further notes and remarks available.

## **2.7.13 Delete configuration assignment from BOM**

### **2.7.13.1 Start state and preconditions**

A CAD.bill\_of\_material with an assigned CONF.configuration exists and is in a state allowing change.

### **2.7.13.2 Required operations**

- Disconnect.CONF.configuration\_bom

### **2.7.13.3 End state and post condition**

A CONF.configuration is no longer assigned to a CAD.bill\_of\_material.

#### **2.7.13.4 Notes and remarks**

The deletion of the relation CONF.configuration\_bom at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.7.14 Add solution to configuration**

#### **2.7.14.1 Start state and preconditions**

A CONF.solution exists. A CONF.configuration exists and is in a state allowing change.

#### **2.7.14.2 Required operations**

- Connect.CONF.configuration\_solution

#### **2.7.14.3 End state and post condition**

An additional CONF.solution is assigned to a CONF.configuration.

#### **2.7.14.4 Notes and remarks**

No further notes and remarks available.

### **2.7.15 Delete solution from configuration**

#### **2.7.15.1 Start state and preconditions**

A CONF.configuration with assigned CONF.solution exists and is in a state allowing change.

#### **2.7.15.2 Required operations**

- Disconnect.CONF.configuration\_solution

#### **2.7.15.3 End state and post condition**

A CONF.solution is no longer assigned to a CONF.configuration.

#### **2.7.15.4 Notes and remarks**

The deletion of the relation CONF.configuration\_solution at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.7.16 Delete configuration**

#### **2.7.16.1 Start state and preconditions**

A CONF.configuration does exist and is in a state allowing it to be deleted, i.e. it has no CAD.bill\_of\_material and/or CONF.solution and/or BASE.analysis\_version assigned. The CONF.configuration might have an associated BASE.administration which is also subject to be deleted then (see chapter 2.1).

#### **2.7.16.2 Required operations**

- Delete.CONF.configuration

And, if the configuration is hierarchic, additionally:

- Disconnect.CONF.configuration\_hierarchy

#### **2.7.16.3 End state and post condition**

The CONF.configuration is no longer available within the data management system.

#### **2.7.16.4 Notes and remarks**

The deletion of CONF.configuration at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.7.17 Assign configuration to analysis**

#### **2.7.17.1 Start state and preconditions**

A BASE.analysis\_version exists. A CONF.configuration exists and both are in a state allowing change.

#### **2.7.17.2 Required operations**

- Connect.CONF.configuration\_analysis\_version

#### **2.7.17.3 End state and post condition**

A CONF.configuration is assigned to a BASE.analysis\_version.

#### **2.7.17.4 Notes and remarks**

No further notes and remarks available.

### **2.7.18 Delete configuration assignment from analysis**

#### **2.7.18.1 Start state and preconditions**

A BASE.model\_version with an assigned CONF.configuration exists and is in a state allowing change.

#### **2.7.18.2 Required operations**

- Disconnect.CONF.configuration\_analysis\_version

#### **2.7.18.3 End state and post condition**

A CONF.configuration is no longer assigned to a BASE.model\_version.

#### **2.7.18.4 Notes and remarks**

The deletion of the relation CONF.configuration\_analysis\_version at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.8 Functionality of use case Model Definition Management**

See SimPDM recommendation for a general description of the use case Model Definition Management.

### **2.8.1 Create new model**

#### **2.8.1.1 Start state and preconditions**

No specific precondition to be defined.

#### **2.8.1.2 Required operations**

- Create.BASE.model
- Create.BASE.model\_version
- Connect.BASE.model\_version

#### **2.8.1.3 End state and post condition**

A new instance of the class BASE.model and an associated initial instance of the class BASE.model\_version for the new instance of the class BASE.model are available.

#### **2.8.1.4 Notes and remarks**

A new instance of the class `BASE.model` can be defined without initially being embedded within a specific analysis scope. The instances of the classes `BASE.model` and `BASE.model_version` are only defined as nodes within the simulation data management system by this function without any topological content defined. Topological content may be defined and associated either by applying the use cases Topological Element Definition Management, Topological Structure Definition Management and Property Definition Management or by applying the use case Document Management. Optionally the use case Administration Management might be applied for `BASE.model` and `BASE.model_version`.

### **2.8.2 Delete existing model**

#### **2.8.2.1 Start state and preconditions**

The instance of the class `BASE.model` to be deleted has not more than one associated instances of the class `BASE.model_version`, the initial instance of the class `BASE.model_version` for the instance of the class `BASE.model` to be deleted. The initial instance of the class `BASE.model_version` has no relationships with other instances of the class `BASE.model_version`. The associated initial instance of the class `BASE.model_version` has no associated topological description. The instance of the class `BASE.model` and the initial instance of the class `BASE.model_version` each might have associated instances of the class `BASE.administration`.

#### **2.8.2.2 Required operations**

- `Disconnect.BASE.model_version`
- `Delete.BASE.model_version`
- `Delete.BASE.model`

And if the model or the related model version have associated administrative data...

- `Disconnect.BASE.model_administration`
- `Disconnect.BASE.model_version_administration`
- `Delete.BASE.administration`

#### **2.8.2.3 End state and post condition**

The instance of the class `BASE.model` and its associated initial instance of the class `BASE.model_version` are no longer available within the data management system.

#### **2.8.2.4 Notes and remarks**

The deletion of instances of the class `BASE.model` at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.8.3 Create new model version**

#### **2.8.3.1 Start state and preconditions**

An instance of the class `BASE.model` with at least one instance of the class `BASE.model_version`, the initial instance of the class `BASE.model_version` for the instance of the class `BASE.model`, exists.

#### **2.8.3.2 Required operations**

- `Create.BASE.model_version`
- `Connect.BASE.model_version`
- `Connect.BASE.model_version_relationship`

### 2.8.3.3 End state and post condition

A new instance of the class `BASE.model_version` of an existing instance of the class `BASE.model` is available within the data management system. The new instance of the class `BASE.model_version` is connected with another instance of the class `BASE.model_version` of the same instance of the class `BASE.model` in a predecessor–successor relationship.

### 2.8.3.4 Notes and remarks

The relationship `BASE.model_version_relationship` can be used to connect two instances of the class `BASE.model_version`. If these two instances of the class `BASE.model_version` belong to the same instance of the class `BASE.version`, the relationship `BASE.model_version_relationship` defines a predecessor–successor relationship of the instances of the class `BASE.model_version`. Optionally the use case Administration Management might be applied for the instance of the class `BASE.model_version`.

## 2.8.4 Delete existing model version

### 2.8.4.1 Start state and preconditions

The instance of the class `BASE.model_version` to be deleted has no relationships with other instances of the class `BASE.model_version`, no related succeeding instance of the class `BASE.model_version`, no associated topological description and no associated instance of the class `BASE.document`. The instance of the class `BASE.model_version` might have an associated instance of the class `BASE.administration`.

### 2.8.4.2 Required operations

- `Disconnect.BASE.model_version_relationship`
- `Disconnect.BASE.model_version`
- `Delete.BASE.model_version`

And if the model version has an associated administrative data...

- `Disconnect.BASE.model_version_administration`
- `Delete.BASE.administration`

### 2.8.4.3 End state and post condition

The instance of the class `BASE.model_version` is no longer available within the data management system.

### 2.8.4.4 Notes and remarks

The deletion of instances of the class `BASE.model_version` at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.8.5 Connect model version to analysis version

### 2.8.5.1 Start state and preconditions

One instance of the class `BASE.analysis_version` and one instance of the class `BASE.model_version` do exist.

### 2.8.5.2 Required operations

- `Connect.BASE.analysis_version_model_version`

### 2.8.5.3 End state and post condition

An instance of the class `BASE.model_version` is assigned to an instance of the class `BASE.analysis_version`.

### 2.8.5.4 Notes and remarks

No further notes and remarks available

## 2.8.6 Disconnect model version from analysis version

### 2.8.6.1 Start state and preconditions

A relationship between an instance of the class `BASE.analysis_version` and an instance of the class `BASE.model_version` do exist.

### 2.8.6.2 Required operations

- `Disconnect.BASE.analysis_version_model_version`

### 2.8.6.3 End state and post condition

The instance of the class `BASE.model_version` is no longer connected with an instance of the class `BASE.analysis_version`.

### 2.8.6.4 Notes and remarks

An instance of the class `BASE.model_version` can be assigned to more than one instances of the class `BASE.analysis_version`. In this case the deletion on one relationship has no effects on other relationships between an instance of the class `BASE.model_version` to other instances of the class `BASE.analysis_version`.

## 2.8.7 Define new model version dependency

### 2.8.7.1 Start state and preconditions

Two instances of the class `BASE.model_version` associated with different instances of the class `BASE.model` do exist.

### 2.8.7.2 Required operations

- `Connect.BASE.model_version_relationship`

### 2.8.7.3 End state and post condition

Two instances of the class `BASE.model_version` of different instances of the class `BASE.model` are associated with each other in a general manner.

### 2.8.7.4 Notes and remarks

The two different instances of the class `BASE.model_version` associated with different instances of the class `BASE.model` are generally associated with each other. The data model defined by the version 2.0 of the recommendation does not provide different kind of model version relationships explicitly. A relationship between instances of the class `BASE.model_version` associated with different instances of the class `BASE.model` are not supposed to represent a preceding – succeeding relationship of instances of the class `BASE.model_version`.

## 2.8.8 Delete existing model version dependency

### 2.8.8.1 Start state and preconditions

A relationship between two instances of the class `BASE.model_version` associated with different instances of the class `BASE.model` does exist.

### 2.8.8.2 Required operations

- `Delete.BASE.model_version_relationship`

### 2.8.8.3 End state and post condition

The dependency between two instances of the class `BASE.model_version` associated with different with different instances of the class `BASE.model` is no longer available.

#### **2.8.8.4 Notes and remarks**

The deletion of the dependency between two instances of the class `BASE.model_version` associated with different instances of the class `BASE.model` has not any effect on the availability and usability of the formally connected instances of the class `BASE.model_version`.

### **2.9 Functionality of use case Output Specification Management**

See SimPDM recommendation for a general description of the use case Output Specification Management.

#### **2.9.1 Create new Output Specification**

##### **2.9.1.1 Start state and preconditions**

A `BASE.analysis_version` exists and is in state allowing change.

##### **2.9.1.2 Required operations**

- `Create.BASE.output_specification`
- `Connect.BASE.output_specification`

And if required...

- `Create.BASE.output_request`
- `Connect.BASE.output_request`

##### **2.9.1.3 End state and post condition**

A `BASE.output_specification` and possibly its `BASE.output_request` is available within the data management system.

##### **2.9.1.4 Notes and remarks**

No further notes and remarks available.

#### **2.9.2 Delete existing Output Specification**

##### **2.9.2.1 Start state and preconditions**

A `BASE.output_specification` exists and is assigned to a `BASE.analysis_version` which is in a state allowing change. Optionally a `BASE.output_request` is assigned to the `BASE.output_specification` and is also in a state allowing change, i.e. no document is assigned.

##### **2.9.2.2 Required operations**

- `Delete.BASE.output_specification`
- `Disconnect.BASE.output_specification`

And if necessary...

- `Delete.BASE.output_request`
- `Disconnect.BASE.output_request`

##### **2.9.2.3 End state and post condition**

A `BASE.output_specification`, its `BASE.output_request` is no longer available and no longer assigned to a `BASE.analysis_version` within the data management system.

##### **2.9.2.4 Notes and remarks**

The deletion of the `BASE.output_specification`, its `BASE.output_request` at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.9.3 Define new group of interesting elements

### 2.9.3.1 Start state and preconditions

A BASE.output\_specification exists and is in a state allowing change. Furthermore TOPO.model\_element or one of its subtypes does exist.

### 2.9.3.2 Required operations

- Create.BASE.group\_of\_elements
- Connect.BASE.output\_group
- Connect.TOPO.element\_of\_interest

### 2.9.3.3 End state and post condition

A BASE.group\_of\_elements is available within the data management system, is assigned to a BASE.output\_specification and has assigned some TOPO.model\_element or one of its subtypes.

### 2.9.3.4 Notes and remarks

No further notes and remarks available.

## 2.9.4 Delete group definition of interesting elements

### 2.9.4.1 Start state and preconditions

A BASE.group\_of\_elements exists and has assigned some TOPO.model\_element or one of its subtypes. BASE.group\_of\_elements is assigned to a BASE.output\_specification which is in a state allowing change.

### 2.9.4.2 Required operations

- Delete.BASE.group\_of\_elements
- Disconnect.BASE.output\_group
- Disconnect.TOPO.element\_of\_interest

### 2.9.4.3 End state and post condition

A BASE.group\_of\_elements and its assignments are no longer available within the data management system.

### 2.9.4.4 Notes and remarks

The TOPO.model\_element or one of its subtypes still exists within the data management system. Deletion is subject of another use case (see chapter 2.15).

The deletion of the BASE.group\_of\_elements at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.10 Functionality of use case Parameter Association Management

See SimPDM recommendation for a general description of the use case Parameter Association Management.

### 2.10.1 Create new parameter

#### 2.10.1.1 Start state and preconditions

One instance of one of the following classes does exist.

- BASE.model\_version
- BASE.transformation

- CAD.connection\_element\_definition
- CAD.connection\_type
- PROP.parameter\_specified\_property
- SETT.setting
- SYNC.parameter\_synchronisation
- TOPO.model\_element

### 2.10.1.2 Required operations

- Create.BASE.parameter (one of the subtypes respectively)

And, depending on the instance of a class, for which the new parameter is to be defined, one of ...

- Connect.BASE.model\_parameter
- Connect.BASE.transformation\_parameter
- Connect.CAD.connection\_element\_definition\_parameter
- Connect.CAD.connection\_type\_parameter
- Connect.PROP.property\_parameter
- Connect.SETT.setting\_parameter
- Connect.SYNC.parameter\_synchronisation\_parameter
- Connect.TOPO.element\_parameter

### 2.10.1.3 End state and post condition

A new instance of the class BASE.parameter (one of the subtypes respectively) exists and is assigned to an instance of one of the classes

- BASE.model\_version
- BASE.transformation
- CAD.connection\_element\_definition
- CAD.connection\_type
- PROP.parameter\_specified\_property
- SETT.setting
- SYNC.parameter\_synchronisation
- TOPO.model\_element

### 2.10.1.4 Notes and remarks

An instance of the class BASE.parameter (one of the subtypes respectively) usually is used once in the context of a certain object. In case of process linking and networking an instance of the class BASE.parameter (one of the subtypes respectively) can be shared by more than one object.

## 2.10.2 Delete existing parameter

### 2.10.2.1 Start state and preconditions

An instance of the class BASE.parameter (one of the subtypes respectively) to be deleted does exist.

### 2.10.2.2 Required operations

- Delete.BASE.parameter (one of the subtypes respectively)

And, depending of the instance of a class for which the parameter is defined, one of ...

- Disconnect.BASE.model\_parameter

- Disconnect.BASE.transformation\_parameter
- Disconnect.CAD.connection\_element\_definition\_parameter
- Disconnect.CAD.connection\_type\_parameter
- Disconnect.PROP.property\_parameter
- Disconnect.SETT.setting\_parameter
- Disconnect.SYNC.parameter\_synchronisation\_parameter
- Disconnect.TOPO.element\_parameter

### 2.10.2.3 End state and post condition

The instance of the class BASE.parameter (one of the subtypes respectively) is no longer available.

### 2.10.2.4 Notes and remarks

The deletion of an instance of the class BASE.parameter (one of the subtypes respectively) at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.10.3 Add a parameter unit

### 2.10.3.1 Start state and preconditions

A BASE.parameter (one of the subtypes respectively) exists and is in state allowing change.

### 2.10.3.2 Required operations

- Create.BASE.unit
- Connect.BASE.parameter\_unit

### 2.10.3.3 End state and post condition

An instance of the class BASE.unit is available within the data management system and is assigned to a BASE.parameter (one of the subtypes respectively).

### 2.10.3.4 Notes and remarks

No further notes and remarks available.

## 2.10.4 Delete a parameter unit

### 2.10.4.1 Start state and preconditions

An instance of the class BASE.unit to be deleted does exist and is in a stage allowing change.

### 2.10.4.2 Required operations

- Delete.BASE.unit
- Disconnect.BASE.parameter\_unit

### 2.10.4.3 End state and post condition

The instance of the class BASE.unit is no longer assigned to a BASE.parameter (one of the subtypes respectively) and is no longer available on the data base. The instance of the class BASE.parameter (one of the subtypes respectively) is still available, for deletion see according chapter above.

### 2.10.4.4 Notes and remarks

The deletion of an instance of the class BASE.unit at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.11 Functionality of use case PDM Information Derivation Management

See SimPDM recommendation for a general description of the use case Geometry Derivation Management.

### 2.11.1 Create new CAD-PDM-Information

#### 2.11.1.1 Start state and preconditions

No start state and precondition to be defined.

#### 2.11.1.2 Required operations

- Create.CAD.cadpdm\_information (or one of the subtypes)

#### 2.11.1.3 End state and post condition

A new CAD.cadpdm\_information definition is available within the data management system.

#### 2.11.1.4 Notes and remarks

Use of CAD.cadpdm\_information is meaningful only in combination with other mechanism, functionalities of use case Geometry Derivation Management like the ones described in the following in this chapter.

### 2.11.2 Delete CAD-PDM-Information

#### 2.11.2.1 Start state and preconditions

A CAD.cadpdm\_information to be deleted does exist and is in a stage allowing change, i.e. it has no further objects like CONF.configuration, BASE.document, etc. assigned. The CAD.cadpdm\_information might have an associated BASE.administration which is also subject to be deleted then (see chapter 2.1).

#### 2.11.2.2 Required operations

- Delete.CAD.cadpdm\_information (or one of the subtypes)

#### 2.11.2.3 End state and post condition

The CAD.cadpdm\_information definition is no longer available within the data management system

#### 2.11.2.4 Notes and remarks

The deletion of a CAD.cadpdm\_information object at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### 2.11.3 Derive CAD-PDM-Information

#### 2.11.3.1 Start state and preconditions

A (specialized) CAD.cadpdm\_information exists.

#### 2.11.3.2 Required operations

- Create.CAD.cadpdm\_information (one of its sub types)

And, depending on the specialization type of CAD.cadpdm\_information, one of...

- Connect.CAD.derived\_shape
- Connect.CAD.derived\_cel
- Connect.CAD.derived\_bom

### **2.11.3.3 End state and post condition**

A derived and (specialized) CAD.cadpdm\_information is available and assigned to its origin within the data management system.

### **2.11.3.4 Notes and remarks**

Derivation is only possible within a type of specialization, not cross-specialization.

## **2.11.4 Delete CAD-PDM-Information derivation**

### **2.11.4.1 Start state and preconditions**

A (specialized) CAD.cadpdm\_information and its origin exist and are in state allowing change.

### **2.11.4.2 Required operations**

- Delete.CAD.cadpdm\_information (one of its sub types)

And, depending on the specialization type of CAD.cadpdm\_information, one of...

- Disconnect.CAD.derived\_shape
- Disconnect.CAD.derived\_cel
- Disconnect.CAD.derived\_bom

### **2.11.4.3 End state and post condition**

A derived and (specialized) CAD.cadpdm\_information is no longer available within the data management system.

### **2.11.4.4 Notes and remarks**

The deletion of derived CAD.cadpdm\_information at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.11.5 Assign BOM to analysis**

### **2.11.5.1 Start state and preconditions**

A CAD.bill\_of\_material does exist. A BASE.analysis\_version exists and is in a stage allowing change.

### **2.11.5.2 Required operations**

- Connect.CAD.analysis\_bom

### **2.11.5.3 End state and post condition**

A CAD.bill\_of\_material is assigned to a BASE.analysis\_version within the data management system.

### **2.11.5.4 Notes and remarks**

No further notes and remarks available.

## **2.11.6 Delete BOM assignment from analysis**

### **2.11.6.1 Start state and preconditions**

A CAD.bill\_of\_material does exist and is assigned to a BASE.analysis\_version which is in a state allowing change.

### **2.11.6.2 Required operations**

- Disconnect.CAD.analysis\_bom

### **2.11.6.3 End state and post condition**

A CAD.bill\_of\_material is no longer assigned to a BASE.analysis\_version within the data management system.

### **2.11.6.4 Notes and remarks**

The deletion of the relation CAD.analysis\_bom at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.11.7 Create BOM parts**

### **2.11.7.1 Start state and preconditions**

A CAD.bill\_of\_material exists and is in a state allowing change. Additionally existence of a PROP.property\_set\_version can be a precondition.

### **2.11.7.2 Required operations**

- Create.CAD.part
- Connect.CAD.bom\_part

And, if necessary and applicable

- Connect.CAD.material\_of\_part

### **2.11.7.3 End state and post condition**

A CAD.part is available and assigned to a CAD.bill\_of\_material within the data management system. Additionally the CAD.part can be assigned to a PROP.property\_set\_version.

### **2.11.7.4 Notes and remarks**

The PROP.property\_set\_version already exists within the data management system. Creation is subject of another use case (see chapter 2.13).

## **2.11.8 Deleting BOM parts**

### **2.11.8.1 Start state and preconditions**

A CAD.part exists and is in a state allowing change, i.e. it has no further objects of BASE.model\_version assigned.

### **2.11.8.2 Required operations**

- Delete.CAD.part
- Disconnect.CAD.bom\_part

And, if necessary,

- Disconnect.CAD.material\_of\_part

### **2.11.8.3 End state and post condition**

A CAD.part is no longer available to a CAD.bill\_of\_material within the data management system.

### **2.11.8.4 Notes and remarks**

The PROP.property\_set\_version still exists within the data management system. Deletion is subject of another use case (see chapter 2.13).

The deletion of a CAD.part at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.11.9 Assign part to model

### 2.11.9.1 Start state and preconditions

A CAD.part does exist. A BASE.model\_version exists and is in a state allowing change.

### 2.11.9.2 Required operations

- Connect.CAD.part\_model\_version

### 2.11.9.3 End state and post condition

A CAD.part is assigned to a BASE.model\_version within the data management system.

### 2.11.9.4 Notes and remarks

No further notes and remarks available.

## 2.11.10 Delete part assignment from model

### 2.11.10.1 Start state and preconditions

A CAD.part does exist and is assigned to a BASE.model\_version which is in a state allowing change.

### 2.11.10.2 Required operations

- Disconnect.CAD.part\_model\_version

### 2.11.10.3 End state and post condition

A CAD.part is no longer assigned to a BASE.model\_version within the data management system.

### 2.11.10.4 Notes and remarks

The deletion of the relation CAD.part\_model\_version at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.11.11 Specify CAE model of shape representation

### 2.11.11.1 Start state and preconditions

A CAD.shape\_representation does exist. A BASE.model\_version exists and is in a state allowing change.

### 2.11.11.2 Required operations

- Connect.CAD.shape\_representation\_model\_version

### 2.11.11.3 End state and post condition

A CAD.shape\_representation is assigned to a BASE.model\_version within the data management system.

### 2.11.11.4 Notes and remarks

Difference between CAD.shape\_representation\_model\_version and CAD.used\_shape is on semantic level. CAD.shape\_representation\_model\_version is used to identify a CAE model belonging to a certain shape in a range of derived shapes. This can be used when multiple CAE models base on a range of derived shapes. CAD.used\_shape identifies the origin shape used by a CAE model.

## 2.11.12 Delete CAE model specification from shape representation

### 2.11.12.1 Start state and preconditions

A CAD.shape\_representation does exist and is assigned to a BASE.model\_version where both are in a state allowing change.

### **2.11.12.2 Required operations**

- Disconnect.CAD.shape\_representation\_model\_version

### **2.11.12.3 End state and post condition**

A CAD.shape\_representation is no longer assigned to a BASE.model\_version within the data management system.

### **2.11.12.4 Notes and remarks**

The deletion of the relation CAD.shape\_representation\_model\_version at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.11.13 Assign shape representation to CAE model**

### **2.11.13.1 Start state and preconditions**

A CAD.shape\_representation exists and is in a state allowing change. A BASE.model\_version exists and is in a state allowing change.

### **2.11.13.2 Required operations**

- Connect.CAD.used\_shape

### **2.11.13.3 End state and post condition**

A BASE.model\_version is assigned to a CAD.shape\_representation within the data management system.

### **2.11.13.4 Notes and remarks**

Difference between CAD.shape\_representation\_model\_version and CAD.used\_shape is on semantic level. CAD.shape\_representation\_model\_version is used to identify a CAE model belonging to a certain shape in a range of derived shapes. This can be used when multiple CAE models base on a range of derived shapes. CAD.used\_shape identifies the origin shape used by a CAE model.

## **2.11.14 Delete shape representation assignment from CAE model**

### **2.11.14.1 Start state and preconditions**

A BASE.model\_version does exist and is assigned to a CAD.shape\_representation where both are in a state allowing change.

### **2.11.14.2 Required operations**

- Disconnect.CAD.used\_shape

### **2.11.14.3 End state and post condition**

A BASE.model\_version is no longer assigned to a CAD.shape\_representation within the data management system.

### **2.11.14.4 Notes and remarks**

The deletion of the relation CAD.used\_shape at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.11.15 Assign connection element list to analysis**

### **2.11.15.1 Start state and preconditions**

A CAD.connection\_element\_list does exist. A BASE.analysis\_version exists and is in a state allowing change.

### **2.11.15.2 Required operations**

- Connect.CAD.analysis\_cel

### **2.11.15.3 End state and post condition**

A CAD.connection\_element\_list is assigned to a BASE.analysis\_version within the data management system.

### **2.11.15.4 Notes and remarks**

No further notes and remarks available.

## **2.11.16 Delete connection element list assignment from analysis**

### **2.11.16.1 Start state and preconditions**

A CAD.connection\_element\_list does exist and is assigned to a BASE.analysis\_version which is in a state allowing change.

### **2.11.16.2 Required operations**

- Disconnect.CAD.analysis\_cel

### **2.11.16.3 End state and post condition**

A CAD.connection\_element\_list is no longer assigned to a BASE.analysis\_version within the data management system.

### **2.11.16.4 Notes and remarks**

The deletion of the relation CAD.analysis\_cel at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.11.17 Create connection element**

### **2.11.17.1 Start state and preconditions**

A CAD.connection\_element\_list exists and is in a state allowing change.

### **2.11.17.2 Required operations**

- Create.CAD.connection\_element
- Create.CAD.connection\_element\_definition
- Create.CAD.connection\_element\_type
- Connect.CAD.cel\_elements
- Connect.CAD.connection\_definition
- Connect.CAD.connection\_type\_definition
- Connect.CAD.connection\_material

### **2.11.17.3 End state and post condition**

A CAD.connection\_element, its CAD.connection\_element\_definition and CAD.connection\_element\_type is available and assigned to a CAD.connection\_element\_list within the data management system.

### **2.11.17.4 Notes and remarks**

The actual parameters of the CAD.connection\_element\_definition are defined by BASE.parameter assigned to the CAD.connection\_element\_definition. See chapter 2.10 for the functionality of parameter association management.

The actual parameters and properties of the CAD.connection\_element\_type are defined by BASE.parameter resp. PROP.property\_set\_version assigned to the CAD.connection\_element\_type. See chapter 2.10 resp. chapter 2.13 for the functionality of parameter association management.

## 2.11.18 Delete connection element

### 2.11.18.1 Start state and preconditions

A CAD.connection\_element, its CAD.connection\_element\_definition and CAD.connection\_element\_type exist and they are assigned to a CAD.connection\_element\_list which is in a state allowing change, i.e. there are no further objects like parameters or properties assigned.

### 2.11.18.2 Required operations

- Delete.CAD.connection\_element
- Delete.CAD.connection\_element\_definition
- Delete.CAD.connection\_element\_type
- Disconnect.CAD.cel\_elements
- Disconnect.CAD.connection\_definition
- Disconnect.CAD.connection\_type\_definition
- Disconnect.CAD.connection\_material

### 2.11.18.3 End state and post condition

A CAD.connection\_element, its CAD.connection\_element\_definition and CAD.connection\_element\_type is no longer available and no longer assigned to a CAD.connection\_element\_list within the data management system.

### 2.11.18.4 Notes and remarks

The actual parameters of the CAD.connection\_element\_definition are defined by BASE.parameter assigned to the CAD.connection\_element\_definition. See chapter 2.10 for the functionality of parameter association management.

The actual parameters and properties of the CAD.connection\_element\_type are defined by BASE.parameter resp. PROP.property\_set\_version assigned to the CAD.connection\_element\_type. See chapter 2.10 resp. chapter 2.13 for the functionality of parameter association management.

The deletion of the CAD.connection\_element, its CAD.connection\_element\_definition and CAD.connection\_element\_type at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.12 Functionality of use case Post-processing Management

See SimPDM recommendation for a general description of the use case Post-processing Management.

### 2.12.1 Create analysis result

#### 2.12.1.1 Start state and preconditions

A BASE.analysis\_version exists and is in a state allowing no more changes.

#### 2.12.1.2 Required operations

- Create.BASE.analysis\_result
- Connect.BASE.analysis\_run\_specific\_analysis\_result

### **2.12.1.3 End state and post condition**

A `BASE.analysis_result` is available and assigned to a `BASE.analysis_version` within the data management system.

### **2.12.1.4 Notes and remarks**

It has to be stressed that there must be explicit traceability between a `BASE.analysis_result` and its source, a `BASE.analysis_version`.

## **2.12.2 Delete analysis result**

### **2.12.2.1 Start state and preconditions**

A `BASE.analysis_result` exists and is in a state allowing change. There are no further `BASE.documents` or `BASE.administration` assigned to the `BASE.analysis_result` (for disconnecting them see the according chapters 2.1 and 2.4).

### **2.12.2.2 Required operations**

- `Delete.BASE.analysis_result`
- `Disconnect.BASE.analysis_run_specific_analysis_result`

### **2.12.2.3 End state and post condition**

The `BASE.analysis_result` and its assignment to a `BASE.analysis_version` are no longer available within the data management system.

### **2.12.2.4 Notes and remarks**

It has to be stressed that there must be explicit traceability between a `BASE.analysis_result` and its source, a `BASE.analysis_result`. This is even more important in cases of actual result existence (as `BASE.analysis_result` is just a folder). In this case deleting a `BASE.analysis_result` means losing the link between the actual results and their `BASE.analysis_version`. Deleting a `BASE.analysis_result` might be suitable in cases of non-converging results.

The deletion of `BASE.analysis_result` at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.12.3 Create a computation output (result first order)**

### **2.12.3.1 Start state and preconditions**

A `BASE.analysis_result` exists and is in a stage allowing change.

### **2.12.3.2 Required operations**

- `Create.BASE.computation_output`
- `Connect.BASE.result_first_order`

### **2.12.3.3 End state and post condition**

A result first order is represented as an instance of `BASE.computation_output` and is assigned to a specific `BASE.analysis_result` within the data management system.

### **2.12.3.4 Notes and remarks**

It has to be stressed that there must be explicit traceability between every result and its source, a `BASE.analysis_version`.

## 2.12.4 Delete a computation output (result first order)

### 2.12.4.1 Start state and preconditions

A BASE.computation\_result exists and is in a stage allowing change.

### 2.12.4.2 Required operations

- Delete.BASE.computation\_output
- Disconnect.BASE.result\_first\_order

### 2.12.4.3 End state and post condition

A result first order represented as an instance of BASE.computation\_output is no longer available within the data management system.

### 2.12.4.4 Notes and remarks

It has to be stressed that there must be explicit traceability between every result and its source, a BASE.analysis\_version.

The deletion of BASE.computation\_output at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.12.5 Create a key result (result second order)

### 2.12.5.1 Start state and preconditions

A BASE.analysis\_result and a BASE.computation\_output or an already derived BASE.key\_result exists.

### 2.12.5.2 Required operations

- Create.BASE.key\_result
- Create.BASE.postprocessing\_input (one of its sub types)
- Connect.BASE:result\_second\_order
- Connect.BASE.base\_for\_key\_result

And, if using a post processing template

- Connect.BASE.template\_creates\_key\_result

### 2.12.5.3 End state and post condition

A result second order is represented as an instance of BASE.key\_result and is assigned to a specific BASE.analysis\_result. Furthermore its derivation chain is represented.

### 2.12.5.4 Notes and remarks

It has to be stressed that there must be explicit traceability between every result and its sources, BASE.analysis\_version and BASE.postprocessing\_input (one of its sub types).

## 2.12.6 Delete a key result (result second order)

### 2.12.6.1 Start state and preconditions

A BASE.key\_result exists and is in a state allowing change.

### 2.12.6.2 Required operations

- Delete.BASE.key\_result
- Disconnect.BASE:result\_second\_order

- Disconnect.BASE.base\_for\_key\_result

And, if using a post processing template

- Disconnect.BASE.template\_creates\_key\_result

### 2.12.6.3 End state and post condition

A result second order represented as an instance of BASE.key\_result is no longer available within the data management system. Its post processing input still exists

### 2.12.6.4 Notes and remarks

It has to be stressed that there must be explicit traceability between every result and its sources, BASE.analysis\_version and BASE.postprocessing\_input (one of its sub types).

The deletion of BASE.key\_result at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.12.7 Create a report (result third order)

### 2.12.7.1 Start state and preconditions

A BASE.analysis\_result exists. Depending on the sourcing for the report, a BASE.key\_result and/or BASE.computation\_result exist.

### 2.12.7.2 Required operations

- Create.BASE.report
- Connect.BASE.result\_third\_order

And, depending of the report sourcing, at least one of ...

- Connect.BASE.base\_for\_report
- Connect.BASE.base\_for\_report2

### 2.12.7.3 End state and post condition

A result third order is represented as an instance of BASE.report and is assigned to a specific BASE.analysis\_result. Furthermore its derivation chain is represented.

### 2.12.7.4 Notes and remarks

A BASE.report can be derived from results first and second order. A BASE.report can belong to different BASE.analysis\_result, as product performance conclusion usually needs more than a single simulation.

It has to be stressed that there must be explicit traceability between every result and its sources, BASE.analysis\_version, BASE.computation\_output and BASE.key\_result.

## 2.12.8 Delete a report (result third order)

### 2.12.8.1 Start state and preconditions

A BASE.report exists and is in a state allowing change.

### 2.12.8.2 Required operations

- Delete.BASE.report
- Disconnect.BASE.result\_third\_order

And, depending of the report sourcing, at least one of ...

- Disconnect.BASE.base\_for\_report
- Disconnect.BASE.base\_for\_report2

### 2.12.8.3 End state and post condition

A result third order represented as an instance of BASE.report is no longer available within the data management system. Its sourcing objects like instances of BASE.key\_result or BASE.computation\_output input still exist.

### 2.12.8.4 Notes and remarks

It has to be stressed that there must be explicit traceability between every result and its sources, BASE.analysis\_version, BASE.computation\_output and BASE.key\_result.

The deletion of BASE.report at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.12.9 Create a detailed post processing object

### 2.12.9.1 Start state and preconditions

A BASE.key\_result or a BASE.template exists and is in a state allowing change.

### 2.12.9.2 Required operations

- Create.POST.detailed\_postprocessing (one of its sub types)

Depending on the type of result which is detailed, one of ...

- Connect.POST.detailing\_key\_result
- Connect.POST.detailing\_report

Depending on the POST.detailed\_postprocessing sub type, one of ...

- Connect.POST.model\_version\_from\_result
- Connect.POST.property\_set\_version\_from\_result
- Connect.POST.cadpdm\_information\_from\_result

### 2.12.9.3 End state and post condition

A detailed post processing objects as an instance of the class POST.detailed\_postprocessing is available within the data management system and assigned to an instance of BASE.key\_result or BASE.template.

### 2.12.9.4 Notes and remarks

Technical definition of the detailed result as an instance of the classes BASE.model\_version, CAD.cadpdm\_information or PROP.property\_set\_version is not subject of this use case. Therefore see chapters 2.8, 2.13 and 2.11.

## 2.12.10 Delete a detailed post processing object

### 2.12.10.1 Start state and preconditions

A POST.detailed\_posprocessing (one of its sub types) exists and is in a state allowing change.

### 2.12.10.2 Required operations

- Delete.POST.detailed\_postprocessing (one of its sub types)

Depending on the type of result which is detailed, one of ...

- Disconnect.POST.detailing\_key\_result
- Disconnect.POST.detailing\_report

Depending on the POST.detailed\_postprocessing sub type, one of ...

- Disconnect.POST.model\_version\_from\_result

- Disconnect.POST.property\_set\_version\_from\_result
- Disconnect.POST.cadpdm\_information\_from\_result

### **2.12.10.3 End state and post condition**

A detailed post processing objects as an instance of the class POST.detailed\_postprocessing is no longer available within the data management system.

The basing instances of the classes BASE.key\_result or BASE.report as well as the technical definition of BASE.model\_version, CAD.cadpdm\_information or PROP.property\_set\_version are still available on the data management system.

### **2.12.10.4 Notes and remarks**

Technical definition and deletion of the detailed result as an instance of the classes BASE.model\_version, CAD.cadpdm\_information or PROP.property\_set\_version is not subject of this use case. Therefore see chapters 2.8, 2.13 and 2.11.

The deletion of POST.detailed\_postprocessing at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.12.11 Create a template**

### **2.12.11.1 Start state and preconditions**

No start state and precondition to be defined.

### **2.12.11.2 Required operations**

- Create.BASE.template

### **2.12.11.3 End state and post condition**

A BASE.template is available within the data management system.

### **2.12.11.4 Notes and remarks**

No further notes and remarks available.

## **2.12.12 Delete a template**

### **2.12.12.1 Start state and preconditions**

A BASE.template exists and is in a state allowing change. There are no further BASE.documents, BASE.key\_result or BASE.administration etc. assigned to the BASE.template (for disconnecting them see the according chapters).

### **2.12.12.2 Required operations**

- Delete.BASE.template

### **2.12.12.3 End state and post condition**

A BASE.template is no longer available within the data management system.

### **2.12.12.4 Notes and remarks**

The deletion of BASE.template at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.13 Functionality of use case Property Definition Management**

See SimPDM recommendation for a general description of the use case Property Definition Management.

## 2.13.1 Create new property set

### 2.13.1.1 Start state and preconditions

No start state and precondition to be defined.

### 2.13.1.2 Required operations

- Create.PROP.property\_set (or its sub type)

### 2.13.1.3 End state and post condition

A new instance of the class PROP.property\_set (or its sub type) is available within the data management system.

### 2.13.1.4 Notes and remarks

No further notes and remarks available.

## 2.13.2 Delete a property set

### 2.13.2.1 Start state and preconditions

An instance of the class PROP.proerty\_set exists and is in a state allowing change. This implies there are no further objects like instances of the class PROP.property\_set\_version, assigned. See the according chapters for deletion.

### 2.13.2.2 Required operations

- Delete.PROP.property\_set (or its sub type)

### 2.13.2.3 End state and post condition

The instance of the class PROP.property\_set is no longer available within the data management system.

### 2.13.2.4 Notes and remarks

The deletion of PROP.property\_set at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.13.3 Create new property set version

### 2.13.3.1 Start state and preconditions

An instance of the class PROP.property\_set does exist.

### 2.13.3.2 Required operations

- Create.PROP.property\_set\_version
- Connect.PROP.property\_set\_version

And optionally

- Connect.PROP.property\_set\_version\_relationship

### 2.13.3.3 End state and post condition

A new instance of the class PROP.property\_set\_version does exist, and is optionally associated with the preceding instance of the class PROP.property\_set\_version in a preceding – succeeding relationship.

### 2.13.3.4 Notes and remarks

Optionally the use case Administration Management might be applied for the new instance of the class PROP.property\_set\_version.

## 2.13.4 Delete existing property set version

### 2.13.4.1 Start state and preconditions

An instance of the class PROP.property\_set\_version does exist. The instance of the class PROP.property\_set\_version belongs to an instance of the class PROP.property\_set and is not used in any relationships. The instance of the class PROP.property\_set\_version might have associated administrative data.

### 2.13.4.2 Required operations

- Delete.PROP.property\_set\_version
- Disconnect.PROP.property\_set\_version

And optionally

- Disconnect.PROP.property\_set\_version\_relationship

### 2.13.4.3 End state and post condition

The instance of the class PROP.property\_set\_version is no longer available within the data management system.

### 2.13.4.4 Notes and remarks

The deletion of instances of the class PROP.property\_set\_version at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.13.5 Create new property set version relationship

### 2.13.5.1 Start state and preconditions

Two instances of the class PROP.property\_set\_version do exist.

### 2.13.5.2 Required operations

- Connect.PROP.property\_set\_version\_relationship

### 2.13.5.3 End state and post condition

Two instances of the class PROP.property\_set\_version are associated with each other in a general manner.

### 2.13.5.4 Notes and remarks

No further notes and remarks available

## 2.13.6 Delete existing property set version relationship

### 2.13.6.1 Start state and preconditions

A relationship between two instances of the class PROP.property\_set\_version to be deleted does exist.

### 2.13.6.2 Required operations

- Disconnect.PROP.property\_set\_version\_relationship

### 2.13.6.3 End state and post condition

The two instances of the class PROP.property\_set\_version are no longer associated with each other.

#### **2.13.6.4 Notes and remarks**

The deletion of instances of the class PROP.property\_set\_version\_relationship at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.13.7 Create new property set structure**

#### **2.13.7.1 Start state and preconditions**

Two instances of the class PROP.property\_set do exist.

#### **2.13.7.2 Required operations**

- Connect.PROP.property\_set\_structure

#### **2.13.7.3 End state and post condition**

Two instances of the class PROP.property\_set are associated with each other in a structure of property sets.

#### **2.13.7.4 Notes and remarks**

A structure of properties can be used to define complex property sets built up by subsets.

### **2.13.8 Delete existing property set structure**

#### **2.13.8.1 Start state and preconditions**

A relationship PROP.property\_set\_structure between two instances of the class PROP.property\_set to be deleted does exist.

#### **2.13.8.2 Required operations**

- Disconnect.PROP.property\_set\_structure

#### **2.13.8.3 End state and post condition**

The two instances of the class PROP.property\_set are no longer associated with each other.

#### **2.13.8.4 Notes and remarks**

The deletion of instances of the relationship PROP.property\_set\_structure at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.13.9 Create a property**

#### **2.13.9.1 Start state and preconditions**

An instance of the class PROP.property\_set\_version exists and is in a state allowing change.

#### **2.13.9.2 Required operations**

- Create.PROP.property (one of its sub types)
- Connect.PROP.property\_of\_set

And, depending on the sub type of PROP.property, possibly one of ...

- Connect.PROP.element\_specification
- Connect.PROP.property\_parameter

#### **2.13.9.3 End state and post condition**

A property as an instance of the class PROP.property is available within the data management system and is assigned to an instance of the class PROP.property\_set\_version.

#### **2.13.9.4 Notes and remarks**

No further notes and remarks available

### **2.13.10 Delete a property**

#### **2.13.10.1 Start state and preconditions**

An instance of the class PROP.property exists and is in a state allowing change. There are no further objects assigned to the property.

#### **2.13.10.2 Required operations**

- Delete.PROP.property (one of its sub types)
- Disconnect.PROP.property\_of\_set

And, depending on the sub type of PROP.property, possibly one of ...

- Disconnect.PROP.element\_specification
- Disconnect.PROP.property\_parameter

#### **2.13.10.3 End state and post condition**

A property as an instance of the class PROP.property is no longer available within the data management system.

#### **2.13.10.4 Notes and remarks**

The deletion of instances of the class PROP.property at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.13.11 Create property relationship**

#### **2.13.11.1 Start state and preconditions**

Two instances of the class PROP.property do exist.

#### **2.13.11.2 Required operations**

- Connect.PROP.property\_relationship

#### **2.13.11.3 End state and post condition**

Two instances of the class PROP.property are associated with each other in a general manner.

#### **2.13.11.4 Notes and remarks**

To create a property hierarchy it can be useful to delete the relationship PROP.property\_of\_set of the lower level property.

### **2.13.12 Delete existing property relationship**

#### **2.13.12.1 Start state and preconditions**

An instance of the relationship PROP.property\_relationship between two instances of the class PROP.property does exist.

#### **2.13.12.2 Required operations**

- Disconnect.PROP.property\_relationship

#### **2.13.12.3 End state and post condition**

Two instances of the class PROP.property are no longer associated with each other.

#### **2.13.12.4 Notes and remarks**

The deletion of instance of the relationship PROP.property\_relationship at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.13.13 Create property constraint**

#### **2.13.13.1 Start state and preconditions**

An instance of the class PROP.property does exist.

#### **2.13.13.2 Required operations**

- Create PROP.property\_constraint
- Connect.PROP.property\_constraint\_property

#### **2.13.13.3 End state and post condition**

An instance of the class PROP.property\_constraint is defined and assigned to an instance of the class PROP.property.

#### **2.13.13.4 Notes and remarks**

Optionally the use case Document Management might be applied for the new instance of the class PROP.property\_constraint.

### **2.13.14 Delete existing property constraint**

#### **2.13.14.1 Start state and preconditions**

An instance of the class PROP.property\_constraint does exist.

#### **2.13.14.2 Required operations**

- Delete PROP.property\_constraint
- Disconnect.PROP.property\_constraint\_property

And if the use case Document Management is applied for the instance of the class PROP.property\_constraint,

- Disconnect.PROP.property\_constraint\_document

#### **2.13.14.3 End state and post condition**

The instance of the class PROP.property\_constraint for an instance of the class PROP.property is no longer available with the data management system

#### **2.13.14.4 Notes and remarks**

The deletion of instances of the class PROP.property\_constraint at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.13.15 Assign property set version to model version**

#### **2.13.15.1 Start state and preconditions**

An instance of the class PROP.property\_set\_version and an instance of the class BASE.model\_version do exist.

#### **2.13.15.2 Required operations**

- Connect.PROP.model\_version\_property\_set\_version

### **2.13.15.3 End state and post condition**

An instance of the class PROP.property\_set\_version is assigned to an instance of the class BASE.model\_version.

### **2.13.15.4 Notes and remarks**

No further note and remark available.

## **2.13.16 Detele property set version from model version**

### **2.13.16.1 Start state and preconditions**

An instance of the relationship PROP.model\_version\_property\_set\_version between an instance of the class PROP.model\_version and an instance of the class PROP property\_set\_version does exist.

### **2.13.16.2 Required operations**

- Disconnect.PROP.model\_version\_property\_set\_version

### **2.13.16.3 End state and post condition**

The instance of the class PROP.property\_set\_version is no longer associated with an instance of the class BASE.model\_version.

### **2.13.16.4 Notes and remarks**

The deletion of instances of the relationship PROP.model\_version\_property\_set\_version at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.13.17 Connect property set version to model element**

### **2.13.17.1 Start state and preconditions**

An instance of a subtype of the class TOPO.model\_element and an instance of the class PROP.property\_set\_version do exist.

### **2.13.17.2 Required operations**

- Connect.PROP.associated\_property\_set\_version

### **2.13.17.3 End state and post condition**

An instance of the class TOPO.model\_element is described by a set of properties.

### **2.13.17.4 Notes and remarks**

No further notes and remarks available.

## **2.13.18 Disconnect property set version to model element**

### **2.13.18.1 Start state and preconditions**

A relationship between an instance of a subtype of the class TOPO.model\_element and an instance of the class PROP.property\_set\_version does exist.

### **2.13.18.2 Required operations**

- Disconnect.PROP.associated\_property\_set\_version

### **2.13.18.3 End state and post condition**

The assignment of the instance of the class PROP.property\_set\_version to an instance of the class TOPO.model\_element is no longer available.

#### **2.13.18.4 Notes and remarks**

No further notes and remarks available.

### **2.14 Functionality of use case Setting Definition Management**

See SimPDM recommendation for a general description of the use case Setting Definition Management.

#### **2.14.1 Create new setting**

##### **2.14.1.1 Start state and preconditions**

No start state and precondition to be defined.

##### **2.14.1.2 Required operations**

- Create.SETT.setting (or one of the subtypes)

##### **2.14.1.3 End state and post condition**

A new SETT.setting definition is available within the data management system.

##### **2.14.1.4 Notes and remarks**

No further notes and remarks available.

#### **2.14.2 Delete existing setting**

##### **2.14.2.1 Start state and preconditions**

A SETT.setting to be deleted does exist and is in a stage allowing change, i.e. it has no further objects like BASE.document, etc. assigned. The SETT.setting might have an associated BASE.administration which is also subject to be deleted then (see chapter 2.1).

##### **2.14.2.2 Required operations**

- Delete. SETT.setting (or one of the subtypes)

##### **2.14.2.3 End state and post condition**

The SETT.setting definition is no longer available within the data management system.

##### **2.14.2.4 Notes and remarks**

The deletion of SETT.setting at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

#### **2.14.3 Assign setting to an analysis**

##### **2.14.3.1 Start state and preconditions**

A BASE.analysis\_version does exist and is in a state allowing change. For high granularity load management existence of SETT.setting or one of its subtypes can be necessary.

##### **2.14.3.2 Required operations**

- Create.BASE.setting\_for\_analysis
- Connect.BASE.setting\_for\_analysis\_version

And, in case of high granularity level data management, additionally

- Create.SETT.applied\_settings (or one of its subtypes)
- Connect.SETT.applied\_settings\_for\_analysis

And, at least one of...

- Connect.SETT.applied\_analysis\_setting
- Connect.SETT.applied\_environment\_setting
- Connect.SETT.applied\_general\_setting
- Connect.SETT.applied\_user\_defined\_sub\_routine

### 2.14.3.3 End state and post condition

A BASE.load\_for\_analysis is assigned to a BASE.analysis\_version. In case of high level granularity this is further detailed in the assigned SETT.applied\_settings and its assignments.

### 2.14.3.4 Notes and remarks

BASE.setting\_for\_analysis can be defined in a document. Document attachment in general is subject of another use case (see chapter 2.4).

## 2.14.4 Delete setting assignment from analysis

### 2.14.4.1 Start state and preconditions

A BASE.setting\_for\_analysis is assigned to a BASE.analysis\_version which is in a state allowing change. In case of high level granularity the setting is further detailed in the assigned SETT.applied\_settings or one of its subtypes and its assignments which also exist and are in a state allowing change. There is no BASE.document attached to BASE.setting\_for\_analysis (for that see chapter 2.4).

### 2.14.4.2 Required operations

- Delete.BASE.setting\_for\_analysis
- Disconnect.BASE.setting\_for\_analysis\_version

And, in case of high granularity level data management, additionally

- Delete.SETT.applied\_settings (or one of its subtypes)
- Disconnect.SETT.applied\_settings\_for\_analysis

And, depending on assigned settings ...

- Disconnect.SETT.applied\_analysis\_setting
- Disconnect.SETT.applied\_environment\_setting
- Disconnect.SETT.applied\_general\_setting
- Disconnect.SETT.applied\_user\_defined\_sub\_routine

### 2.14.4.3 End state and post condition

A BASE.setting\_for\_analysis is no longer available within the data management system. Also no longer available are the SETT.applied\_setting or one of its subtypes and the assignments.

### 2.14.4.4 Notes and remarks

The SETT.setting or one of its subtypes and the BASE.model\_version still exist within the data management system.

The deletion of setting assignments at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.14.5 Create analysis step

### 2.14.5.1 Start state and preconditions

A BASE.setting\_for\_analysis exists and is in a state allowing change. A LOAD.applied\_load\_case also exists, is also in a state allowing change and is assigned (indirectly) to the same BASE.analyse\_version.

### 2.14.5.2 Required operations

- Create.SETT.analysis\_step
- Connect.SETT.applied\_settings\_for\_analysis
- Connect.SETT.analysis\_step\_applied\_load

### 2.14.5.3 End state and post condition

A SETT.analysis\_step organises as a setting the sequence of load cases within a BASE.analysis\_version. This organisation is available within the data management system.

### 2.14.5.4 Notes and remarks

It is important that assigned objects (settings as well as loads) belong to the same BASE.analysis\_version.

## 2.14.6 Delete analysis step

### 2.14.6.1 Start state and preconditions

A SETT.analysis\_step exists and is in a state allowing change. There are no further BASE.documents, BASE.parameters or BASE.administration assigned to the SETT.analysis\_step (for disconnecting them see the according chapters).

### 2.14.6.2 Required operations

- Delete.SETT.analysis\_step
- Disconnect.SETT.applied\_settings\_for\_analysis
- Disconnect.SETT.analysis\_step\_applied\_load

### 2.14.6.3 End state and post condition

A SETT.analysis\_step is no longer available within the data management system.

### 2.14.6.4 Notes and remarks

The BASE.settings\_for\_analysis and the LOAD.applied\_load\_case still exist within the data management system (for deletion refer to the according chapters).

The deletion of SETT.analysis\_step at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.14.7 Add another detailed setting to an analysis

### 2.14.7.1 Start state and preconditions

A BASE.setting\_for\_analysis does exist and is in a state allowing change. A SETT.setting or one of its subtypes exists.

### 2.14.7.2 Required operations

- Create.SETT.applied\_settings (or one of its subtypes)
- Connect.SETT.applied\_settings\_for\_analysis

And one of...

- Connect.SETT.applied\_analysis\_setting
- Connect.SETT.applied\_environment\_setting
- Connect.SETT.applied\_general\_setting
- Connect.SETT.applied\_user\_defined\_sub\_routine

### 2.14.7.3 End state and post condition

Another SETT.applied\_settings or one of its subtypes and its defining SETT.setting or one of its subtypes is assigned to BASE.settings\_for\_analysis within the data management system.

### 2.14.7.4 Notes and remarks

No further notes and remarks available.

## 2.14.8 Delete a detailed setting assignment

### 2.14.8.1 Start state and preconditions

A SETT.applied\_settings or one of its subtypes and its defining SETT.setting or one of its subtypes is assigned to BASE.settings\_for\_analysis within the data management system. The SETT.applied\_settings is in a state allowing change.

### 2.14.8.2 Required operations

- Delete.SETT.applied\_settings (or one of its subtypes)
- Disconnect.SETT.applied\_settings\_for\_analysis

And depending on the assigned setting ...

- Disconnect.SETT.applied\_analysis\_setting
- Disconnect.SETT.applied\_environment\_setting
- Disconnect.SETT.applied\_general\_setting
- Disconnect.SETT.applied\_user\_defined\_sub\_routine

### 2.14.8.3 End state and post condition

A SETT.applied\_settings or one of its subtypes is no longer available at the data management systems. Its relations are also no longer available.

### 2.14.8.4 Notes and remarks

The deletion of setting assignments at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.15 Functionality of use case Topology Element Definition Management

See SimPDM recommendation for a general description of the use case Topology Element Definition Management.

### 2.15.1 Create new topological element

#### 2.15.1.1 Start state and preconditions

No start state and precondition to be defined.

#### 2.15.1.2 Required operations

- Create.TOPO.model\_element (or one of the subtypes)

### **2.15.1.3 End state and post condition**

A new instance of the class TOPO.model\_element definition is available.

### **2.15.1.4 Notes and remarks**

A topological element should either be assigned to an instance of the class BASE.model (use case Topology Structure Definition Management involved) or be managed by a topological element library. Instances of topological elements should not exist unconnected and unmanaged within a data management system. If a topology element library is not available or a topology element is created independently from the library, the new topology element should be assigned to an instance of the class BASE.model\_version.

Optionally the use cases Document Management, Administration Management and Parameter Association Management might be applied.

## **2.15.2 Delete existing topological element**

### **2.15.2.1 Start state and preconditions**

An instance of the class TOPO.model\_element to be deleted does exist.

### **2.15.2.2 Required operations**

- Delete.TOPO.model\_element (or one of the subtypes)

### **2.15.2.3 End state and post condition**

The instance of the class TOPO.model\_element definition is no longer available within the data management system.

### **2.15.2.4 Notes and remarks**

Actually, the deletion of instances of the class TOPO.model\_element from a library is not a daily business, but rather an administrative task of a library manager.

The deletion of instances of the class TOPO.model\_element at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.15.3 Create topological element relationship**

### **2.15.3.1 Start state and preconditions**

At least two instances of the class TOPO.model\_element do exist.

### **2.15.3.2 Required operations**

- Connect.TOPO.model\_element\_relationship

### **2.15.3.3 End state and post condition**

Two instances of the class TOPO.model\_element are related with each other in a general manner.

### **2.15.3.4 Notes and remarks**

No further notes and remarks available

## **2.15.4 Delete topological element relationship**

### **2.15.4.1 Start state and preconditions**

A relationship between two instances of the class TOPO.model\_element does exist.

### **2.15.4.2 Required operations**

- Disconnect.TOPO.model\_element\_relationship

### **2.15.4.3 End state and post condition**

The relationship between two instances of the class TOPO.model\_element is no longer available.

### **2.15.4.4 Notes and remarks**

The deletion of instances of the relationship TOPO.model\_element\_relationship at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## **2.15.5 Include model element into model**

### **2.15.5.1 Start state and preconditions**

At least one instance of the class BASE.model\_version and one instance of the class TOPO.model\_element do exist.

### **2.15.5.2 Required operations**

- Connect.TOPO.of\_model

### **2.15.5.3 End state and post condition**

A model element is included into a model.

### **2.15.5.4 Notes and remarks**

No further notes and remarks available

## **2.15.6 Delete model element from model**

### **2.15.6.1 Start state and preconditions**

At least one instance of the class BASE.model\_version with at least one included instance of the class TOPO.model\_element or one of its sub classes does exist.

### **2.15.6.2 Required operations**

- Disconnect.TOPO.of\_model

### **2.15.6.3 End state and post condition**

A TOPO:model\_element is no longer included into a model, but is still available within the data management system.

### **2.15.6.4 Notes and remarks**

An instance of the class TOPO.model\_element should not exist outside of a model, if it is not managed by a topology element library. Therefore it should automatically be deleted.

## **2.16 Functionality of use case Topology Structure Definition Management**

See SimPDM recommendation for a general description of the use case Topology Structure Definition Management.

### **2.16.1 Define interface element**

#### **2.16.1.1 Start state and preconditions**

An instance of one of the classes TOPO.geo\_element, TOPO.node or TOPO.surface or of one their sub classes does exist.

### 2.16.1.2 Required operations

- Create.TOPO.model\_interface

And, depending on the type of model element, that is to be defined as an interface element, one of...

- Connect.TOPO.surface\_model\_interface
- Connect.TOPO.node\_model\_interface
- Connect.TOPO.geo\_element\_model\_interface

### 2.16.1.3 End state and post condition

A specific model element within a topological structure is specified as an interface element.

### 2.16.1.4 Notes and remarks

Model elements specified as interface elements can be used for automatic topological connections between different models.

## 2.16.2 Delete interface element

### 2.16.2.1 Start state and preconditions

An instance of the class TOPO.model\_interface to be deleted does exist and is not used by a relationship with another instance of the class TOPO.model\_interface.

### 2.16.2.2 Required operations

- Delete.TOPO.model\_interface

And, depending on the type of model element, that is defined as an interface element, one of...

- Disconnect.TOPO.surface\_model\_interface
- Disconnect.TOPO.node\_model\_interface
- Disconnect.TOPO.geo\_element\_model\_interface

### 2.16.2.3 End state and post condition

A specific model element within a topological structure is no longer specified as an interface element.

### 2.16.2.4 Notes and remarks

The instance of the class TOPO.model\_element that was formerly specified as an interface element is still available.

The deletion of instances of the class TOPO.model\_interface and the relationships at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

## 2.16.3 Create topological model connection

### 2.16.3.1 Start state and preconditions

Two instances of the class TOPO.interface\_element belonging to different instances of the class BASE.model do exist.

### 2.16.3.2 Required operations

- Connect.TOPO.cross\_model\_interface\_relation

### 2.16.3.3 End state and post condition

Two different instances of the class BASE.model\_version are connected by an instance of the class TOPO.interface\_element in each of the model versions represented by instances of the class BASE.model\_version.

#### **2.16.3.4 Notes and remarks**

An instance of the class TOPO.interface\_element can only once at a time be used in a topological model connection.

### **2.16.4 Delete topological model connection**

#### **2.16.4.1 Start state and preconditions**

A relationship between two instances of the class TOPO.interface\_element belonging to different instances of the class BASE.model does exist.

#### **2.16.4.2 Required operations**

- Disconnect.TOPO.cross\_model\_interface\_relation

#### **2.16.4.3 End state and post condition**

The two instances of the class TOPO.interface\_element are no longer connected.

#### **2.16.4.4 Notes and remarks**

The deletion of instances of the relationship TOPO.cross\_model\_interface\_relation at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.

### **2.16.5 Assign a surface to a set**

#### **2.16.5.1 Start state and preconditions**

At least one instance of the class TOPO.surface and an instance of the class TOPO.set do exist.

#### **2.16.5.2 Required operations**

- Connect.TOPO.belongs\_to

#### **2.16.5.3 End state and post condition**

The instance of the class TOPO.surface is assigned to an instance of the class TOPO.set.

#### **2.16.5.4 Notes and remarks**

No further notes and remarks do exist.

### **2.16.6 Delete a surface from a set**

#### **2.16.6.1 Start state and preconditions**

At least one instance of the class TOPO.surface assigned to an instance of the class TOPO.set do exist.

#### **2.16.6.2 Required operations**

- Disconnect.TOPO.belongs\_to

#### **2.16.6.3 End state and post condition**

The instance of the class TOPO.surface is no longer assigned to an instance of the class TOPO.set.

#### **2.16.6.4 Notes and remarks**

The instance of the class TOPO.surface formerly assigned to the instance of the class TOPO.set is still available.

The deletion of instances of the relationship TOPO.belongs\_to at this point is described with regards to the data model specification. Nevertheless, the deletion of objects might specifically be restricted by data management system restrictions.