SSP Traceability

Document version: 0.1

January 17, 2020~~December 20, 2019~~

This document is the working draft of the simulation task quality and traceability documentation concept standard (short SSP Traceability). This title is a working title.

This document is a working draft

Document history table

Until the first public release of the document as version 1.0, only those versions will be included in this table, which were made available outside the authoring team, for example for review or to implement the concept in pilot projects.

Version	Date	Remarks
0.1	2019-12-19	First working draft

Abstract

tbd

Conventions used in this document

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as described in RFC 2119 [RFC2119].

Non-normative text is given in square brackets in italic font: [*especially examples are defined in this style.*].

- “Shall”, or the term "REQUIRED", means that the definition or statement is an absolute requirement.
- “Should”, or the adjective "RECOMMENDED", means that there may be valid reasons under certain circumstances for ignoring a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- “May”, or the adjective "OPTIONAL", means that an item is truly optional. It often has the character of a best practice.

Contents

1 Introduction

During the development of products, systems and components, there is very often a need to make design decisions, for example to compare design concepts, to preselect system components or to investigate the system behavior and the interaction of a system with the system environment.

Usually an engineering project initiates and requests a simulation and addresses a simulation order to a simulation department or to a simulation engineer. The results of the simulation then might be, among others, basis for a design decision.

Due to the fact that design decisions might determine the design of a product, a system or a component and determine or significantly influence further product development processes and costs, such design decisions must be based on a reliable information basis with a high degree of trust, especially if functional safety is affected and if the criticality of the simulation is rated high.

This can only be achieved if the underlying simulation tasks have been planned, carried out, secured and documented in a transparent, comprehensible and traceable manner and if there is a high degree of confidence in the correctness and validity of the simulation results.

This can only be achieved if the underlying simulation tasks have been planned, prepared, carried out, validated and documented in a transparent and comprehensible manner and if there is a high degree of confidence in the correctness and validity of the simulation results.

This is the core purpose of the metadata for simulation tasks and the "SSP Traceability" concept approach. The metadata for simulation tasks records all information available for the simulation tasks. This is on the one hand the information handed over by the engineering project along with the simulation order to the simulation department or the simulation engineer and on the other hand all information having been used and generated during the planning, preparation, execution, validation and documentation of the simulation task, including references to externally defined and documented resources (external file or database references) and information on responsibilities, quality checks and approvals.

The metadata of a simulation is stored in XML files called STMD files or glue particle (STMD: Simulation Task Meta Data). These STMD files can be used to inspect and trace simulation tasks (traceability), or to reuse a simulation task or parts of a simulation task (reusability). They may also be used to describe simulation tasks that are assigned to service providers or delegated to development partners (data exchange).

Chapter XXX describes the general approach for the STMD files and the design of the referenced normative STMD XML schema. Chapter XXX describes the structure of STMD files in detail. Chapter XXX documents rules and recommendations for the handling of STMD files, i.e. the creation and use of STMD files. Chapter XXX specifies meaningful resp. recommended tool based functionalities that might support STMD handling use cases. Chapter XXX illustrates the handling and interpretation of STMD files using a simple and easy to understand instantiation example for a DC motor simulation. Finally, Chapter XXX provides a glossary of terms. Die Beschreibungen der Kapitelinhalte müssen von Zeit zu Zeit aktualisiert bzw. zum Schluss ausformuliert werden.

hat formatiert: Deutsch (Deutschland)

2 Glue Particle Traceability Description Layout

The STMD files are based on a generic traceability documentation layout, which determines which descriptive elements are required to document engineering activities in general in a transparent, understandable and comprehensible way in terms of traceability. This layout is basically *independent* of the actual engineering activities described. This means that it can be applied to mechanical design as well as to software development, E/E development, system development and simulation processes.

A glue particle documents a structured scope of engineering activities. The size of this scope depends on the amount and complexity of the activities described within a glue particle. The documentation of such scopes can also be done in several referencing/referenced glue particles.

The traceability documentation layout and its overall structure and descriptive elements will be introduced and explained in the following.

Note:

The description of the traceability documentation layout is completely independent from STMD.

2.1 Overall structure

2.1.1 General information

Each glue particle always has a chapter reserved for information valid for the entire glue particle, i.e. all engineering activities described by a single glue particle.

2.1.2 Technical content structure

The actual technical content is organized in a generic three-level hierarchy structure. This structure contains engineering activity phase's information on the first level and engineering activity step's information on the second level (see figure). The structure and the naming of the phases and steps is derived from the process of the engineering activity described. Each step is broken down into a generic structure with a subchapter for inputs, procedure, outputs, rationales and lifecycle information valid for the step. Life cycle information might also be considered valid for an entire phase which implies, that the life cycle Information is valid for all underlying steps of the phase. In this case life cycle information is documented on the second technical content structure level.

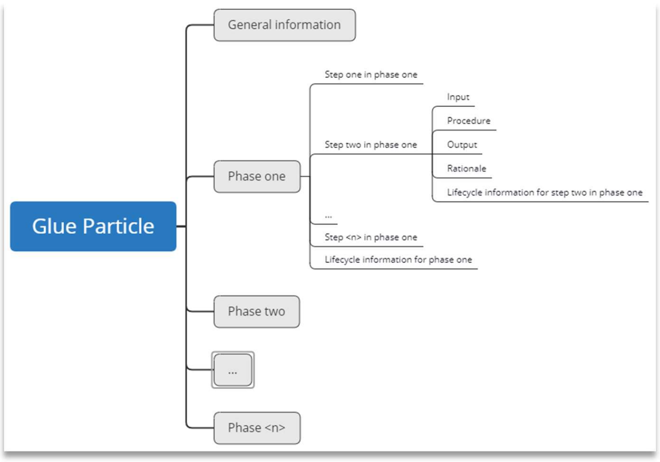


Figure 1: Generic three level structure of a Glue Particle

2.2 Descriptive elements

2.2.1 Title

A glue particle and a structural segment on the first and the second level (phases and steps) within a glue particle has a title. A glue particle can be recognized by its title. However, the title does not serve to uniquely identify a glue particle.

2.2.2 Identifier

A glue particle and a structural segment (XML Element) within a glue particle has an identifier by which a glue particle or the structural segment can be uniquely identified within a given namespace. In principle, an identifier can also be multi-part, e.g. a combination of two attributes or the specification of a given namespace and an ID attribute.

2.2.3 Short description

A short description in a glue particle and a structural segment within a glue particle serves to quickly recognize what the glue particle is documenting without inspecting the contents in detail~~s~~. This can be, for example, a short textual description of the contents or a set of descriptive attributes by which you can recognize the contents or the scope and purpose of a glue particle.

2.2.4 Long description

A long description in a glue particle and a structural segment within a glue particle is the documentation of the actual technical content of a glue particle. It is thus at the same time the legitimation of the glue particle. Without a long description, the glue particle may lose its value or must inevitably be considered incomplete.

2.2.5 Life cycle information

Life cycle information in a glue particle and a structural segment within a glue particle can be used to control processes, but also to assess the formal status of the information of a glue particle. Life cycle Information are life cycle stage specific and comprises the life cycle stage itself, responsibilities, signatures and checksums.

2.2.5.1 Life cycle stage
The life cycle stage itself marks the progress and the liability of an assessed information, i.e. engineering activity phase information and engineering activity step information.

2.2.5.2 Responsible
The responsible is, depending on the life cycle stage either the person who created or edited the information (e.g. for life cycle stage “defined”) or the person who set the life cycle (e.g. for life cycle stage “approved”).

2.2.5.3 Signature
The signature helps to prove that the person who is recorded responsible is actually authentic.

2.2.5.4 Checksum
The checksum should make it possible to determine whether or not the content of a glue particle was changed after the checksum was created by generating a new checksum at a given point in time and comparing it with the checksum stored in the glue particle. If these two checksums are equal, it can be assumed that the glue particle or the checked area of the glue particle is unchanged.

2.2.6 Quality information
Quality information in a glue particle or a structural segment within a glue particle gives a statement about how high the quality of the information and the confidence in the quality of the information is. Unlike the life cycle stage which is only a single label, the quality information can provide more details about the actual quality.

2.2.7 Administrative metadata
These are usually fixed attributes that can be assigned a limited selection of values. Metadata is used both to control data management processes and to search for glue particles.

2.2.8 Key words or classifications
Keywords or classification usually are terms that can be used to describe content, but do not necessarily have to be predefined.

2.2.9 Internal and external object links / references
Internal links serve to avoid or reduce documentation redundancy within a single glue particle file. External links are used to link information that is distributed across several glue particles and to integrate or reference other external resources, such as specifications or procedural instructions.

3 General STMD Approach

3.1 Process view vs. information view

The basis of the STMD approach is the assumption that the planning, preparation, execution, validation and documentation of a simulation task follows an established phase-oriented process that resembles a value-added chain with individual value-added stages from the point of view of simulation. According to this assumed analogy, at the end of each phase in the process, a new value-added stage would be reached, characterized by a set of result types. This means that each phase generates phase-typical results, the existence of which is necessary for reaching the appropriated value-added stage, but not necessarily sufficient. Each phase can be divided into one or more work steps, which can be processed sequentially or in parallel and which themselves produce typical results.

Each process phase and each work step is linked to characteristic information, such as the inputs used, the outputs generated, procedures (i.e. processes and methods) used and also the rationales for procedures used or assumptions made and outputs generated. This information or references to this information are the essential technical metadata of the simulation task, which is stored in the STMD files.

The structure of an STMD file is oriented on the established process for processing a simulation task, but does not represent the process view, but the information view, i.e. the view on the information linked to the process.

3.2 Generic Approach Applied for Simulation Task Meta Data

Although the structure of the STMD files is generally oriented to the established process with its phases and work steps, sufficient flexibility should be maintained in the mapping of the information to be able to describe the large variety of different types of simulation tasks. The simulation for the selection of a DC motor may have completely other metadata than for example the simulation of a velocity control in automated driving and other metadata than the simulation of a BUS communication. The structure of the STMD schema therefore allows sufficient leeway to accommodate all these different metadata.

Therefore, the technical metadata of the individual work steps in the phases are merely divided into subchapters Inputs, Procedure, Outputs and Rationales. Within these subchapters, however, the information can be structured freely to a large extent.

In addition, information can also be documented in external files for which the SSP Traceability Concept does not provide any specifications. References to such external documentation can be used anywhere within the subchapters Inputs, Procedure, Outputs and Rationales. The generic SSP Traceability approach merely specifies how these external resources are referenced (see chapter 3.6.2).

3.3 Version History Information

An STMD file is to be filled stepwise during the planning, preparation, execution and validation of the simulation task so that a version history results, similar to the version history of an office document or a CAD model. However, the history is not documented within single STMD file, but is formed by a sequence of STMD files that represent a version chain. Version management for the metadata of a simulation task is therefore not defined by the STMD XML Schema, but is managed by tools or data management systems. This corresponds approximately to the version mechanism of PDM / TDM systems for the administration of CAD data. The view into a single STMD file therefore does not explicitly reveal which STMD file is the predecessor file in the version chain (is this true?) and it is also not directly obvious which have been the last changes of the STMD file. This can only be determined by a manually or tool-supported comparison with the predecessor file. Each time a glue particle is changed and saved, a new STMD file is created, which then continues the existing version chain by one instance.

Question: Is the attribute <stmd:SimulationTaskMetaData fileversion=""> designed to carry the explicit version ID on an STMD file within a version chain?

Answer: Yes, it is.

To-Do: Intended use or Glue Particle to be described more explicit in document preface/introduction. Example use case to be provided.

hat formatiert: Schriftart: Fett

hat formatiert: Schriftart: Fett

3.4 Key wording, classification and annotations

To make it easier to find STMD files and to facilitate the reuse of STMD files, the SSP Traceability concept approach provides for the use of key wording. A keyword entry consists of a keyword name and a keyword value.

A distinction can be made between mandatory keyword entries with a name defined in the schema and a fixed selection of permitted values, mandatory keyword entries with a name defined in the schema but without a fixed selection of permitted values, and free keyword entries for which both the name and the value can be freely defined.

For example, to specify a keyword to determine the operating system, a keyword entry may be defined with the keyword name *Operating System* and a keyword value like e.g. Windows 10, Windows 7, any Linux and UNIX operating system or even a real time operating system~~s~~.

Similarly, a keyword with the fixed keyword name *Modelled Object* could be defined, whose valid keyword value are not predefined and could b, for example, DC Motor.

As a third example one could define a keyword with the non-predefined keyword name *Context* and the non-predefined keyword value window lifter.

To search for STMD files or simulation tasks that simulate DC motors for window regulators on a Windows 10 system, you could define and execute an appropriate query with the following structure

SEARCH ALL STMD FILES WITH THE KEYWORD CONSTRAINTS

- OPERATING SYSTEM = "WINDOWS 10."
- MODELLED OBJECT = "DC MOTOR"
- CONTEXT = "WINDOW LIFTER"

Keywords can be defined on all structural levels of an STMD file and are referred to as Classification Entry and summarized as Classification.

Annotation may be used for additional free comments and can be defined on all structural levels of an STMD file.

3.5 Lifecycle Information

Lifecycle information can be found both in the subchapter associated with the work steps and in the chapters on the phases. Several lifecycle information can be specified within the chapters so that the lifecycle history of a chapter or subchapter can be documented without resorting to the entire STMD version history. Lifecycle information documented at the chapter level is valid for all subchapters, regardless of the lifecycle information documented in the subchapters.

The lifecycle information refers to the stage of individual sections of the STMD file (chapters for phases and work steps) or their contents. The core of the lifecycle information is the respective stage of the information and, optionally, information about the responsibility for the relevant information or the responsibility for setting the life cycle stage. Also optionally, the stage can be extended with a signature and a checksum for the relevant section.

The available stages are Drafted, Defined, Validated, Approved, Archived, Retracted.

3.6 References

Two different types of references can be used in STMD. Internal references and external references. In the case of the external references, one can distinguish between the referenced types of information and the location of the referenced information, that is, the destination of the external references. Der letze Satz ist noch etwas komisch.

Kommentiert [CF1]: Nach Kap. 3.6.2 verschieben.

3.6.1 Internal References

An internal reference is a reference from one location within the STMD file to another location within the same STMD file. The most common use case of an internal reference is a reference from one work step to the result of another work step in a previous phase. For example, the results of a design specification could be referenced to specify the inputs for a model implementation. On the one hand, this avoids redundancies and on the other hand, it improves the traceability.

3.6.2 External References

The second type of references are references to external resources. External resources include models (Functional Mockup Units - FMU) in a simulation data management system, parameter sets in a parameter database, specification documents in a file system, requirements in a requirements management system, and others.

To-Do: Include some examples for external references

4 STMD Structure

This chapter intents to describe the generic structure of STMD files in details. As already introduced in Chapter 3.2, STMD files are hierarchal segmented in a generic three level structure according the simulation task process where the upper generic level is considered *phase* (also referenced as level -1) and the lower generic level *step* (also referenced as level -2). Each step is further subdivided into seven generic subchapter (also referenced as level -3).

- Inputs
- Procedure
- Outputs
- Rationales
- LifeCycleInformation
- Classification
- Annotations

4.1 Generic three level STMD structure

Phases can be interpreted as components of a process that must be processed in a certain sequence, since the output or the work results of a phase may in turn be the input of a subsequent phase or the prerequisite for its processing.

The seven generic subchapters are the same for all steps in an STMD file. While *Inputs*, Procedure, *Outputs* and *Rationale* can be used in a wide range of flexibility to document technical simulation task related content considered for traceability intentions, *LifeCycleInformation* is an administrative subchapter to document the life cycle stage history of a chapter resp. subchapter information with related responsibilities, signatures and Checksums. *Classification* and *Annotation* allow for defining keywords and adding some additional notes and comments. The four subchapter Inputs, Procedure, Outputs and Rationales are actually the long description as introduced in chapter 2.2.4.

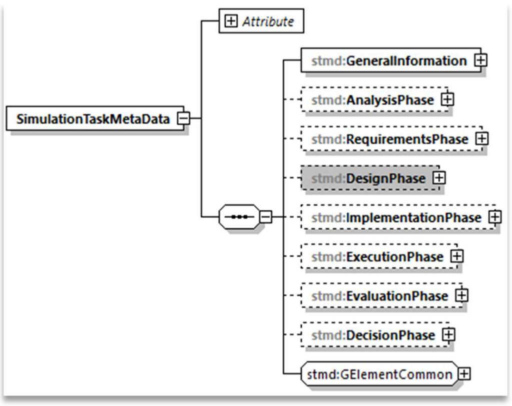


Figure 2: Phases of an STMD file (level -1)

Steps, on the other hand, can be processed sequentially or in parallel, depending on the nature of the simulation task.

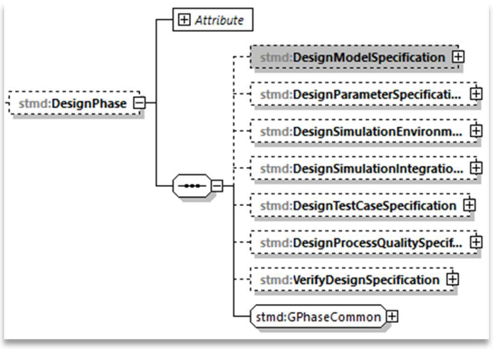

Attribute
stmd:DesignPhase
stmd:DesignModelSpecification
stmd:DesignParameterSpecificati...
stmd:DesignSimulationEnvironm...
stmd:DesignSimulationIntegratio...
stmd:DesignTestCaseSpecification
stmd:DesignProcessQualitySpecif...
stmd:VerifyDesignSpecification
stmd:GPhaseCommon

Figure 3: Steps of an STMD File (level -2; by the example of Design Phase)

The seven generic subchapters are the same for all steps in an STMD file. While *Inputs*, Procedure, *Outputs* and *Rationale* can be used in a wide range of flexibility to document technical simulation task related content considered for traceability intentions, *LifeCycleInformation* is an administrative subchapter to document life cycle related responsibilities, signatures and Checksums and *Classification* and *Annotation* allow for defining keywords and adding some additional notes and comments.

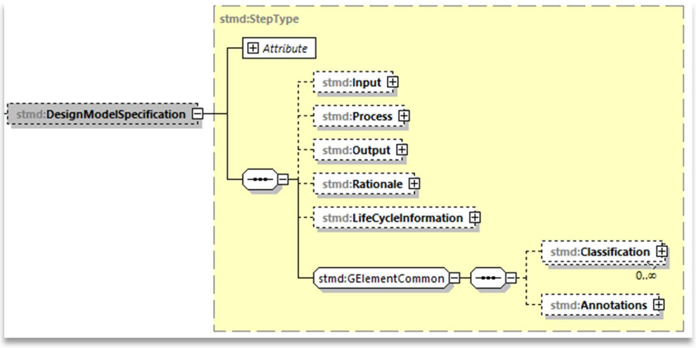


Figure 4: Generic structure of an STMD step (by the example of Design Model Specification) ***(To be replaced)***

4.2 General information

tbd

4.2.1 Derivation chain

tbd

4.3 Analysis Phase

The analysis phase serves to fully understand the engineering task from which a simulation order is generated. This also includes recognizing and understanding for which purpose and with which goal the simulation is commissioned and to understanding the quality requirements for the simulation execution and the simulation results.

4.3.1 Analyse engineering task

An essential step of the analysis phase is the analysis of the superior engineering task, especially with regard to the purpose of the simulation, the objectives associated with the execution of the simulation as well as with regard to the quality requirements for the simulation execution and the simulation results.

4.3.1.1 Inputs

Collection of all input information used to analyse the higher-level engineering task. This can be, for example, a simulation order or information about the higher-level engineering task such as product data, e-mails, meeting minutes or requirement documents.

4.3.1.2 Procedure

Documentation of the procedure for analysing the higher-level engineering task. The procedure could include, for example, inspection of the information and documents provided or meetings to coordinate the simulation task. Reference could also be made to a procedural instruction.

4.3.1.3 Output

Collection of all results of the analysis of the superior engineering task. Results could, for example, be recorded in the form of meeting minutes. A simulation order confirmed by both sides could also be the result of the analysis.

4.3.1.4 Rationale

Reasons for the chosen procedure for the analysis of the superior engineering task and for the results of the analysis. In the rationales, for example, it could be documented why the simulation described in the simulation order is basically suitable to answer the original question from the higher-level engineering task. If specifications or restrictions for the simulation task have already been formulated and agreed, these can also be reasoned at this point.

4.3.1.5 Life cycle information (for analysing the engineering task)

In this chapter, you can store information that documents the status of the documentation for analyzing the engineering task. See chapter 2.2.5 and subchapters for a general description of the lifecycle information.

4.3.2 Verify engineering task analysis

tbd

4.3.2.1 Inputs

Collection of all information used to qualitatively secure the analysis of the higher-level engineering task. Usually, reference is made to the analysis of the higher-level engineering task.

4.3.2.2 Procedure

Documentation of the procedure to secure the quality of the engineering task analysis. Possibly a reference is made to procedural instructions for compliance with quality specifications.

4.3.2.3 Output

Essential content of this chapter is the verdict of the quality assurance for the analysis of the superior engineering task, possibly just an *OK* statement.

4.3.2.4 Rationale

tbd

4.3.2.5 Life Cycle Information (for verifying the engineering task analysis)

In this chapter, information can be stored that documents the status of the documentation for verifying the engineering task analysis. See chapter 2.2.5 and subchapters for a general description of the lifecycle information.

4.3.3 Life Cycle Information (for the entire analysis phase)

In this chapter, information can be stored that documents the status of the documentation for verifying the engineering task analysis. See chapter 2.2.5 and subchapters for a general description of the lifecycle information.

Life cycle information stored in this chapter is valid for all subchapters, independent of the life cycle information stored in subchapters. This means that life cycle information stores in this chapter override life cycle information stored in chapchapter, if so.

4.4 Requirements phase

tbd

4.4.1 Derive model requirements

4.4.2 Derive parameter requirements

tbd

4.4.3 Derive simulation environment requirements

tbd

4.4.4 Derive simulation integration requirements

tbd

4.4.5 Derive test case requirements

tbd

4.4.6 Derive process quality requirements

tbd

4.4.7 Verify requirements

tbd

4.5 Design phase

tbd

4.5.1 Design model specification

tbd

4.5.2 Design parameter specification

tbd

4.5.3 Design simulation environment specification

tbd

4.5.4 Design simulation integration specification

tbd

4.5.5 Design test case specification

tbd

4.5.6 Design process quality specification

tbd

4.5.7 Verify design specification

tbd

4.6 Implementation phase

tbd

4.6.1 Implement model

tbd

4.6.2 Implement parameter

tbd

4.6.3 Implement simulation environment

tbd

4.6.4 Implement test case

tbd

4.6.5 Integrate simulation

tbd

4.6.6 Assure simulation setup quality

tbd

4.6.7 Derive simulation setup quality

tbd

4.7 Execution phase

tbd

4.7.1 Execute simulation

tbd

4.8 Evaluation Phase

4.8.1 Evaluate Simulation Results

tbd

4.8.2 Assure Simulation Quality

tbd

4.8.3 Derive Simulation Quality Verdict

tbd

4.9 Fulfillment Phase

tbd

4.9.1 Derive Objective Fulfilment

tbd

5 Details of the STMD Schema structure

The structure of an STMD file is specified in an STMD XML Schema. The following chapters describe the structure and the associated attribute of all XML elements of STMD files.

Some of the Figure in this chapter are currently out of date and are to be replaced

5.1 SimulationTaskMetaData

The SimulationTaskMetaData element is the all enclosing top EXM element.

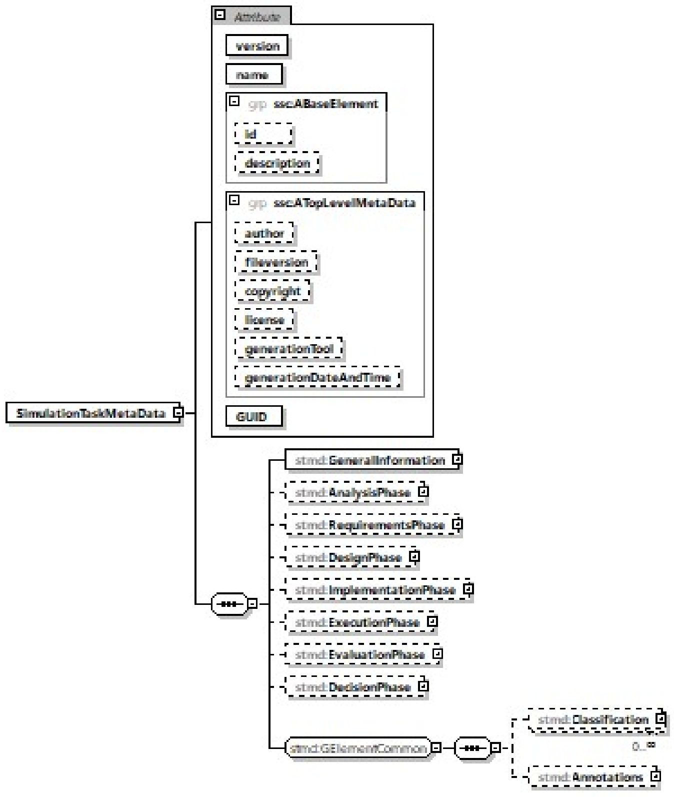


Figure 5: SimulationTaskMetaData structure and attributes (durch ein Bild mit besserer Auflösung ersetzen)

The SimulationTaskMetaData element is structured by subordinated element as described below.

Element name	
SimulationTaskMetaData	
Sub element name	Optional/mandatory
GeneralInformation	mandatory

AnalysisPhase	optional
RequirementsPhase	optional
DesignPhase	optional
ImplementationPhase	optional
ExecutionPhase	optional
EvaluationPhase	optional
FulfilmentPhase	optional

Table 6: Sub elements of SimulationTaskMetaData

Each entry elements is associated with the following attributes as described in the table below.

Element name		
SimulationTaskMetaData		
Elements node XPath (if available)		
stmd:SimulationTaskMetaData		
Attribute name	Optional/mandatory	Attribute description
version	mandatory	Version of STMD format, 0.x for this pre-release.
name	mandatory	This attribute gives the simulation task a name, which can be used for purposes of presenting the simulation task to the user, e.g. when selecting individual variant STMDs from an SSP.
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.
author	optional	This attribute gives the name of the author of this file's content.
fileversion	optional	This attribute gives a version number for this file's content.
Copyright	optional	This attribute gives copyright information for this file's content.
license	optional	This attribute gives license information for this file's content.
generationTool	optional	This attribute gives the name of the tool that generated this file.

generationDateAndTime	optional	This attribute gives the date and time this file was generated.

Figure 7: SimulationTaskMetaData attributes

5.1.1 GeneralInformation

This element is used to encapsulate general information about the simulation task, which is not part of any specific phase or step.

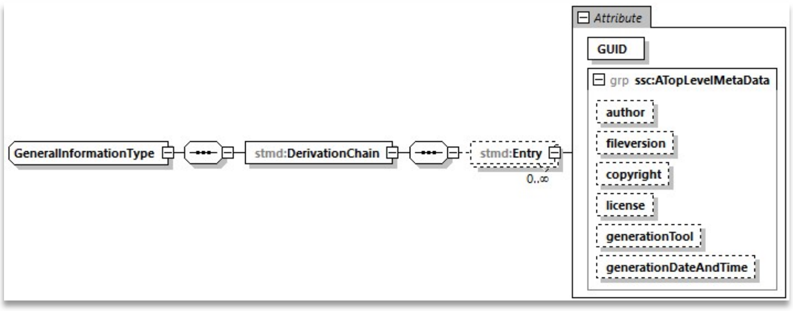


Figure 8: GeneralInformationType structure and attributes ***(To be replaced)***

The GeneralInformationType is structured by subordinated element as described in the table below.

Element name	
GeneralInformationType	
Sub element name	Optional/mandatory
DerivationChain	mandatory

Table 1: Sub elements of GeneralInformationType

The GeneralInformationType is not associated with any attributes.

5.1.1.1 DerivationChain

This element can be used to provide the set of file information that was used to derive the current file. I.e. if the content of the current file can be considered to be derived from one or a set of other STMD files, then the top level meta data and DerivationChain information of those files should be included in their original order as entries in this file's DerivationChain element.

The derivation chain described by the DerivationChain element may contain one or more Entries (with their associated attributes) or even no entry at all.

The DerivationChainEntry elements are not structured by subordinated elements.

Each DerivationChainEntry elements is associated with the following attributes as described in the table below.

Element name
DerivationChainEntry
Elements node XPath (if available)

//element(*,stmd:GeneralInformationType)/stmd:DerivationChain/stmd:DerivationChainEntry/@GUID		
Attribute name	Optional/mandatory	Attribute description
author	optional	This attribute gives the name of the author of this file's content.
fileversion	optional	This attribute gives a version number for this file's content.
copyright	optional	This attribute gives copyright information for this file's content.
license	optional	This attribute gives license information for this file's content.
generationTool	optional	This attribute gives the name of the tool that generated this file.
generationDateAndTime	optional	This attribute gives the date and time this file was generated.

Table 2: DerivationChainEntry attributes

5.1.2 AnalysisPhase

This element specifies the Analysis Phase of the overall Simulation Task.

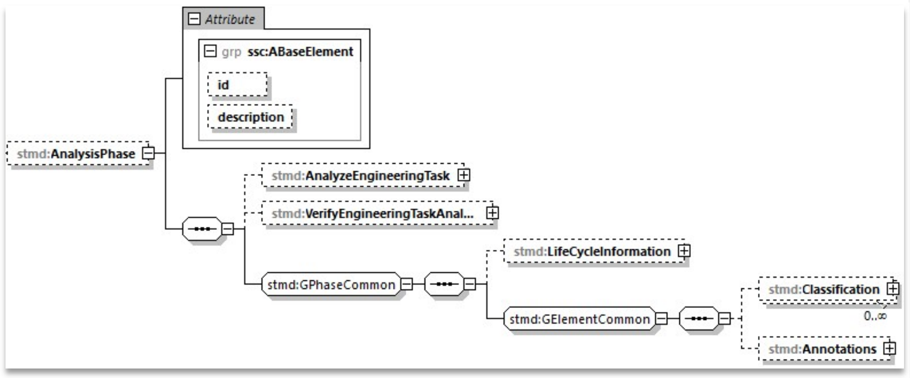


Figure 9: AnalysisPhase structure and attributes

The AnalysisPhase element is structured by subordinated element as listed in the table below.

Element name	
AnalysisPhase	
Sub element name	Optional/mandatory
AnalyzeEngineeringTask	optional
VerifyEngineeringTaskAnalysis	optional
LifeCycleInformation	optional
Classification	optional

Annotations	optional

Table 3: Sub elements of AnalysisPhase

The AnalysisPhase is associated with the following attributes as listed in the table below.

Element name		
AnalysisPhase		
Elements node XPath (if available)		
stmd:SimulationTaskMetaData/stmd:AnalysisPhase		
Attribute name	Optional/mandatory	Attribute description
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 4: AnalysisPhase attributes

5.1.2.1 *AnalyzeEngineeringTask*
For the details of AnalysisEngineeringTask structure and attributes see chapt. 5.2 StepType.

5.1.2.2 *VerifyEngineeringTaskAnalysis*
For the details of VerifyEngineeringTaskAnalysis structure and attributes see chapt. 5.2 StepType.

5.1.2.3 *LifeCycleInformation*
For the details of LifeCycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.1.2.4 *Classification*
For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.1.2.5 *Annotations*
For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.1.3 RequirementsPhase
This element specifies the Requirements Phase of the overall Simulation Task.

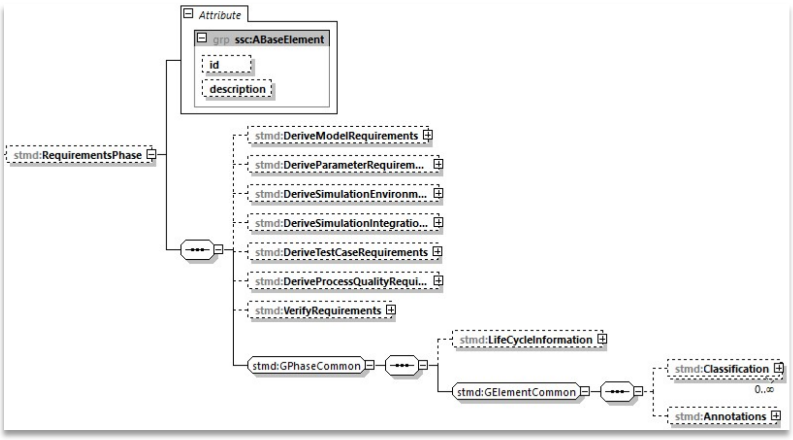

Attribute
grp ssc:ABaseElement
id
description
stmd:RequirementsPhase
stmd:DeriveModelRequirements
stmd:DeriveParameterRequirem...
stmd:DeriveSimulationEnvironm...
stmd:DeriveSimulationIntegratio...
stmd:DeriveTestCaseRequirements
stmd:DeriveProcessQualityRequi...
stmd:VerifyRequirements
stmd:GPhaseCommon
stmd:LifeCycleInformation
stmd:GElementCommon
stmd:Classification
0..∞
stmd:Annotations

Figure 10: RequirementsPhase structure and attributes

The RequirememtsPhase element is structured by subordinated element as listed in the table below.

Element name	
RequirementsPhase	
Sub element name	Optional/mandatory
DeriveModelRequirements	optional
DeriveParameterRequirements	optional
DeriveSimulationEnvironmentRequirements	optional
DeriveSimulationIntegrationRequirements	optional
DeriveTestCaseRequirements	optional
DeriveProcessQualityRequirements	optional
VerifyRequirements	optional
LifeCycleInformation	optional
Classification	optional
Annotations	optional

Table 5: Sub elements of RequirementsPhase

The RequirementsPhase is associated with the following attributes as listed in the table below.

Element name		
RequirementsPhase		
Elements node XPath (if available)		
stmd:SimulationTaskMetaData/stmd:RequirementsPhase		
Attribute name	Optional/mandatory	Attribute description

id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 6: RequirementsPhase attributes

5.1.3.1 DeriveModelRequirements
For the details of DeriveModelRequirements structure and attributes see chapt. 5.2 StepType.

5.1.3.2 DeriveParameterRequirements
For the details of DeriveParameterRequirements structure and attributes see chapt. 5.2 StepType.

5.1.3.3 DeriveSimulationEnvironmentRequirements
For the details of DeriveSimulationEnvironmentRequirememts structure and attributes see chapt. 5.2 StepType.

5.1.3.4 DeriveSimulationIntegrationRequirements
For the details of DeriveSimulationIntegrationRequiremenmts structure and attributes see chapt. 5.2 StepType.

5.1.3.5 DeriveTestCaseRequirements
For the details of DeriveTestCaseRequirements structure and attributes see chapt. 5.2 StepType.

5.1.3.6 DeriveProcessQualityRequirements
For the details of DeriveProcessQualityRequirements structure and attributes see chapt. 5.2 StepType.

5.1.3.7 VerifyRequirements
For the details of VerifyRequirements structure and attributes see chapt. 5.2 StepType.

5.1.3.8 LifeCycleInformation
For the details of LifeCycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.1.3.9 Classification
For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.1.3.10 Annotations
For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.1.4 DesignPhase
This element specifies the Specification Phase of the overall Simulation Task.

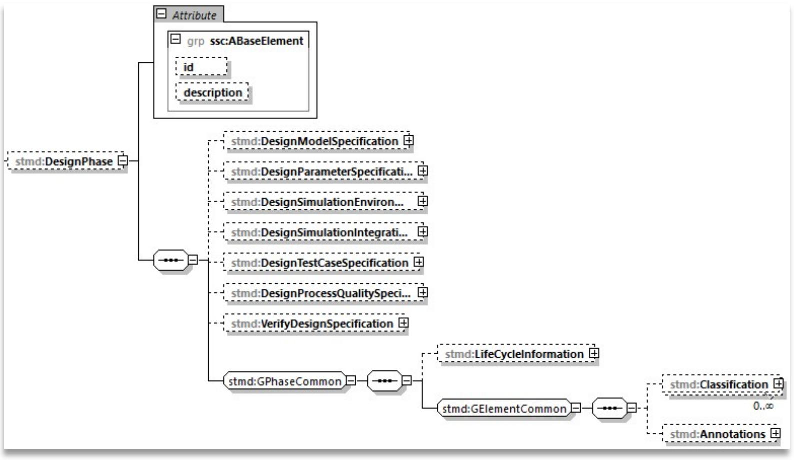

Attribute
grp ssc:ABaseElement
id
description
stmd:DesignPhase
stmd:DesignModelSpecification
stmd:DesignParameterSpecificati...
stmd:DesignSimulationEnviron...
stmd:DesignSimulationIntegrati...
stmd:DesignTestCaseSpecification
stmd:DesignProcessQualitySpeci...
stmd:VerifyDesignSpecification
stmd:GPhaseCommon
stmd:LifeCycleInformation
stmd:GElementCommon
stmd:Classification
0..∞
stmd:Annotations

Figure 11: DesignPhase structure and attributes

The DesignPhase element is structured by subordinated element as listed in the table below.

Element name	
DesignPhase	
Sub element name	Optional/mandatory
DesignModelSpecification	optional
DesignParameterSpecification	optional
DesignSimulationEnvironmentSpecification	optional
DesignSimulationIntegrationSpecification	optional
DesignTestCaseSpecification	optional
DeriveProcessQualityRequirements	optional
VerifyDesignSpecification	optional
LifeCycleInformation	optional
Classification	optional
Annotations	optional

Table 7: Sub elements of DesignPhase

The DesignPhase is associated with the following attributes as listed in the table below.

Element name		
DesignPhase		
Elements node XPath (if available)		
stmd:SimulationTaskMetaData/stmd:DesignPhase		
Attribute name	Optional/mandatory	Attribute description

id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 8: DesignPhase attributes

5.1.4.1 DesignModelSpecification
For the details of DesignModelSpecification structure and attributes see chapt. 5.2 StepType.

5.1.4.2 DesignParameterSpecification
For the details of DesignParameterSpecification structure and attributes see chapt. 5.2 StepType.

5.1.4.3 DesignSimulationEnvironmentSpecification
For the details of DesignSimulatinoEnvironmentSpecification structure and attributes see chapt. 5.2 StepType.

5.1.4.4 DesignSimulationIntegrationSpecification
For the details of DesignSimulationInterationSpecification structure and attributes see chapt. 5.2 StepType.

5.1.4.5 DesignTestCaseSpecification
For the details of DesignTestCaseSpecification structure and attributes see chapt. 5.2 StepType.

5.1.4.6 DeriveProcessQualityRequirements
For the details of DeriveProcessQualityRequirements structure and attributes see chapt. 5.2 StepType.

5.1.4.7 VerifyDesignSpecification
For the details of VerifyDesignSpecification structure and attributes see chapt. 5.2 StepType.

5.1.4.8 LifeCycleInformation
For the details of LifeCycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.1.4.9 Classification
For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.1.4.10 Annotations
For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.1.5 ImplementationPhase
This element specifies the Implementation and Validation Phase of the overall Simulation Task.

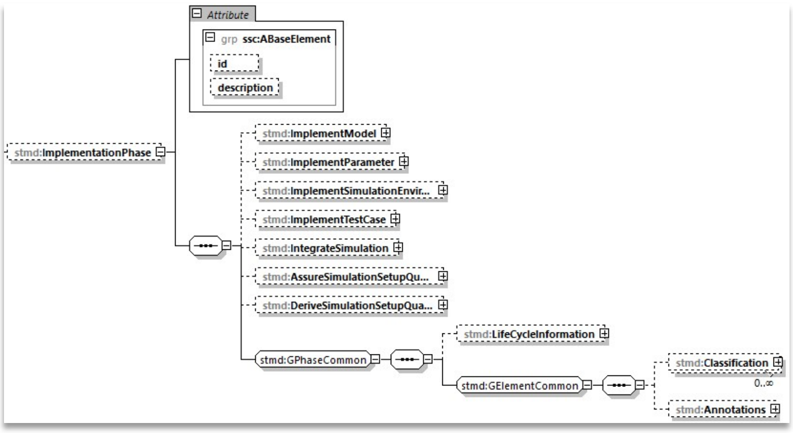


Figure 12: ImplementationPhase structure and attributes

The ImplementationPhase element is structured by subordinated element as listed in the table below.

Element name	
ImplementationPhase	
Sub element name	Optional/mandatory
ImplementModel	optional
ImplementParameter	optional
ImplementSimulationEnvironment	optional
IntegrateSimulation	optional
ImplementTestCase	optional
AssureSimulationSetupQuality	optional
DeriveSimulationSetupQualityVerdict	optional
LifeCycleInformation	optional
Classification	optional
Annotations	optional

Table 9: Sub elements of ImplementationPhase

The ImplementationPhase is associated with the following attributes as listed in the table below.

Element name		
ImplementationPhase		
Elements node XPath (if available)		
stmd:SimulationTaskMetaData/stmd: ImplementationPhase		
Attribute name	Optional/mandatory	Attribute description

id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 10: ImplementationPhase attributes

5.1.5.1 ImplementModel
For the details of ImplementModel structure and attributes see chapt. 5.2 StepType.

5.1.5.2 ImplementParameter
For the details of ImplementParameter structure and attributes see chapt. 5.2 StepType.

5.1.5.3 ImplementSimulationEnvironment
For the details of ImplementSimulationEnvironment structure and attributes see chapt. 5.2 StepType.

5.1.5.4 IntegrateSimulation
For the details of IntegrateSimulation structure and attributes see chapt. 5.2 StepType.

5.1.5.5 ImplementTestCase
For the details of ImplementTestCase structure and attributes see chapt. 5.2 StepType.

5.1.5.6 AssureSimulationSetupQuality
For the details of AssureSimulationSetupQuality structure and attributes see chapt. 5.2 StepType.

5.1.5.7 DeriveSimulationSetupQualityVerdict
For the details of DeriveSimulationSetupQualityVerdict structure and attributes see chapt. 5.2 StepType.

5.1.5.8 LifeCycleInformation
For the details of LifeCycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.1.5.9 Classification
For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.1.5.10 Annotations
For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.1.6 ExecutionPhase
This element specifies the Execution Phase of the overall Simulation Task.

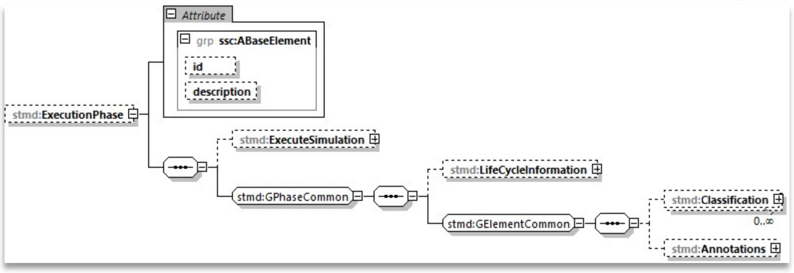


Figure 13: ExecutionPhase structure and attributes

The ExecutionPhasePhase element is structured by subordinated element as listed in the table below.

Element name	
ExecutionPhase	
Sub element name	Optional/mandatory
ExecuteSimulation	optional
LifeCycleInformation	optional
Classification	optional
Annotations	optional

Table 11: Sub elements of ExecutionPhase

The ExecutionPhase is associated with the following attributes as listed in the table below.

Element name		
ExecutionPhase		
Elements node XPath (if available)		
stmd:SimulationTaskMetaData/stmd: ExecutionPhase		
Attribute name	Optional/mandatory	Attribute description
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 12: ExecutionPhase attributes

5.1.6.1 ExecuteSimulation

For the details of ExecuteSimulation structure and attributes see chapt. 5.2 StepType.

5.1.6.2 LifeCycleInformation

For the details of LifeCycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.1.6.3 Classification

For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.1.6.4 Annotations

For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.1.7 EvaluationPhase

This element specifies the Evaluation Phase of the overall Simulation Task.

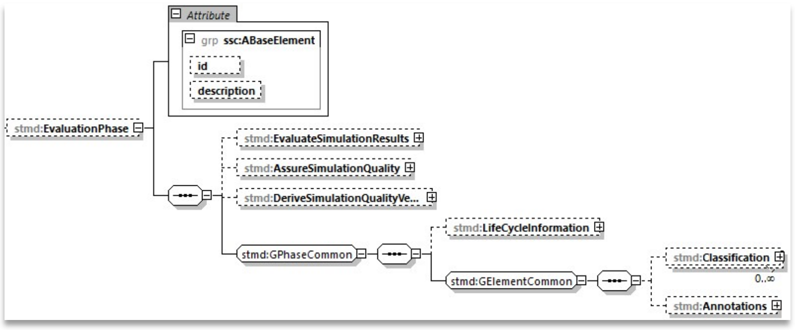


Figure 14: EvaluationPase structure and attributes

The EvaluationPhase element is structured by subordinated element as listed in the table below.

Element name	
EvaluationPhase	
Sub element name	Optional/mandatory
EvaluateSimulationResults	optional
AssureSimulationQuality	optional
DeriveSimulationQualityVerdict	optional
LifeCycleInformation	optional
Classification	optional
Annotations	optional

Table 13: Sub elements of EvaluationPhase

The EvaluationPhase is associated with the following attributes as listed in the table below.

Element name
EvaluationPhase
Elements node XPath (if available)
stmd:SimulationTaskMetaData/stmd: EvaluationPhase

Attribute name	Optional/mandatory	Attribute description
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 14: EvaluationPhase attributes

5.1.7.1 EvaluateSimulationResults
For the details of EvaluateSimulationResults structure and attributes see chapt. 5.2 StepType.

5.1.7.2 AssureSimulationQuality
For the details of AssureSimulationQuality structure and attributes see chapt. 5.2 StepType.

5.1.7.3 DeriveSimulationQualityVerdict
For the details of DeriveSimulationQualityVerdict structure and attributes see chapt. 5.2 StepType.

5.1.7.4 LifeCycleInformation
For the details of LifeCycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.1.7.5 Classification
For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.1.7.6 Annotations
For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.1.8 FulfilmentPhase
This element specifies the Fulfillment Phase of the overall Simulation Task.

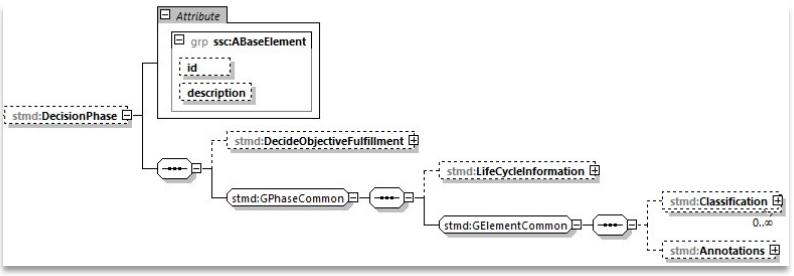


Figure 15: FulfilmentPhase structure and attributes ***(To be replaced)***

The FulfilmentPhase element is structured by subordinated element as listed in the table below.

Element name	
FulfilmentPhase	
Sub element name	Optional/mandatory

DecideObjectiveFulfilment	optional
LifeCycleInformation	optional
Classification	optional
Annotations	optional

Table 15: Sub elements of FulfilmentPhase

The FulfilmentPhase is associated with the following attributes as listed in the table below.

Element name		
FulfilmentPhase		
Elements node XPath (if available)		
stmd:SimulationTaskMetaData/stmd: FulfilmentPhase		
Attribute name	Optional/mandatory	Attribute description
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier.
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 16: FulfilmentPhase attributes

5.1.8.1 DecideObjectiveFulfilment

For the details of DecideObjectiveFulfilment structure and attributes see chapt. 5.2 StepType.

5.1.8.2 LifeCycleInformation

For the details of LifeCycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.1.8.3 Classification

For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.1.8.4 Annotations

For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.2 StepType

This type specifies the content of each individual step inside a phase of the overall Simulation Task.

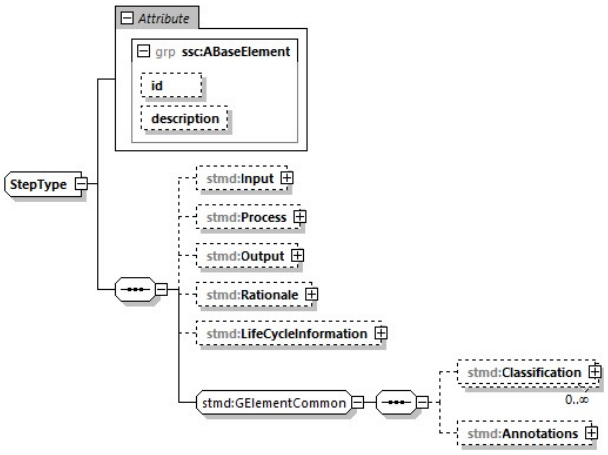


Figure 16: StepType structure and attributes ***(To be replaced)***

The StepType element is structured by subordinated elements as described in the table below.

Element name	
StepType	
Sub element name	Optional/mandatory
Input	optional
Procedure	optional
Output	optional
Rationale	optional
LifeCycleInformation	optional
Classification	optional
Annotations	optional

Table 17: Sub elements of StepType

The StepType is associated with the following attributes as described in the table below.

Element type name		
StepType		
Elements node XPath (if available)		
Attribute name	Optional/mandatory	Attribute description
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier

description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 18: StepType attributes

5.2.1 Input

For the details of Input structure and attributes see chapt. 5.3 ParticleType.

5.2.2 Procedure

For the details of Procedure structure and attributes see chapt. 5.3 ParticleType.

5.2.3 Output

For the details of Output structure and attributes see chapt. 5.3 ParticleType.

5.2.4 Rationale

For the details of Rationale structure and attributes see chapt. 5.3 ParticleType.

5.2.5 LifeCycleInformation

For the details of LifeycleInformation structure and attributes see chapt. 5.4 LifeCycleInformationType.

5.2.6 Classification

For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.2.7 Annotations

For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.3 ParticleType

This type specifies the content of each individual particle inside a step of a phase of the overall Simulation Task.

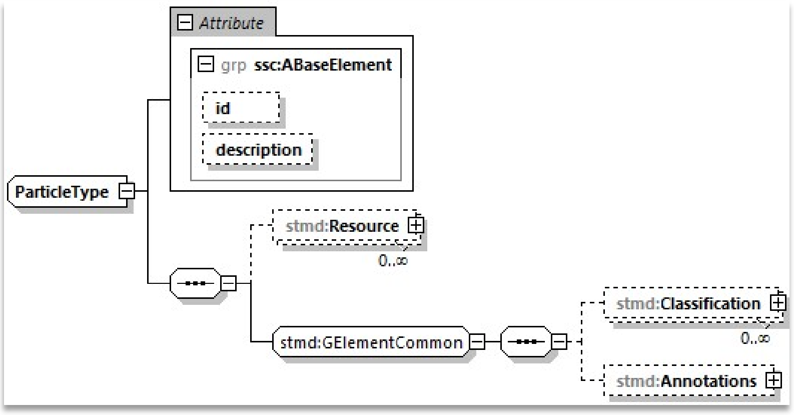


Figure 19: ParticleType structure and attributes

The ParticleType element is structured by subordinated elements as described in the table below.

Element name	
ResourceType	
Sub element name	Optional/mandatory
Resource	optional
Classification	optional
Annotations	optional

Table 20: Sub elements of ParticleType

The ParticleType is associated with the following attributes as described in the table below.

Element type name		
ResourceType		
Elements node XPath (if available)		
Attribute name	Optional/mandatory	Attribute description
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 21: ParticleType attributes

5.3.1 Resource

For the details of Resource structure and attributes see chapt. 5.8 ResourceType.

5.3.2 Classification

For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.3.3 Annotations

For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.4 LifeCycleInformationType

This element contains life cycle information about the enclosing phase or step element. Due to the inherent dependencies of life cycles, life cycle information at later phases will be dependent on life cycle status of former phases to a certain extent: For example, if the Implementation Phase is designated as having reached the status "Validated", it would create a contradiction if the Requirements Phase has only reached status "Drafted". Multiple life cycle information entries can be present, in order to record the historical progression of the life cycle status, however only the last entry in document order, which will also be of the highest maturity, will be considered valid for the current file contents, earlier states only recording historical data.

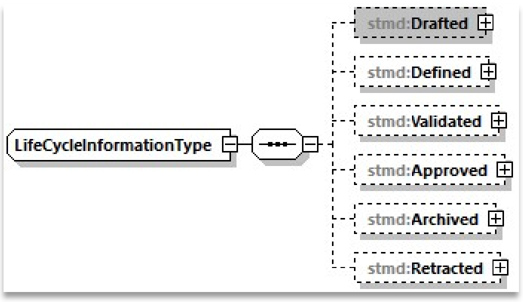


Figure 17: LifecycleInformationType structure

The LifeCycleInformationType is structured by subordinated elements as described in the table below.

Element name	
LifeCycleInformationType	
Sub element name	Optional/mandatory
Drafted	optional
Defined	optional
Validated	optional
Approved	optional
Archived	optional
Retracted	optional

Table 22: Sub elements of LifeCycleEntryType

The ContentType is not associated with any attributes.

5.4.1 Drafted
For details of life cycle entries such as Drafted see chapt. 5.5 LifeCycleEntryType.

5.4.2 Defined
For details of life cycle entries such as Defined see chapt. 5.5 LifeCycleEntryType.

5.4.3 Validated
For details of life cycle entries such as Validated see chapt. 5.5 LifeCycleEntryType.

5.4.4 Approved
For details of life cycle entries such as Approved see chapt. 5.5 LifeCycleEntryType.

5.4.5 Archived
For details of life cycle entries such as Archived see chapt. 5.5 LifeCycleEntryType.

5.4.6 Retracted

For details of life cycle entries such as Retracted see chapt. 5.5 LifeCycleEntryType.

5.5 LifeCycleEntryType

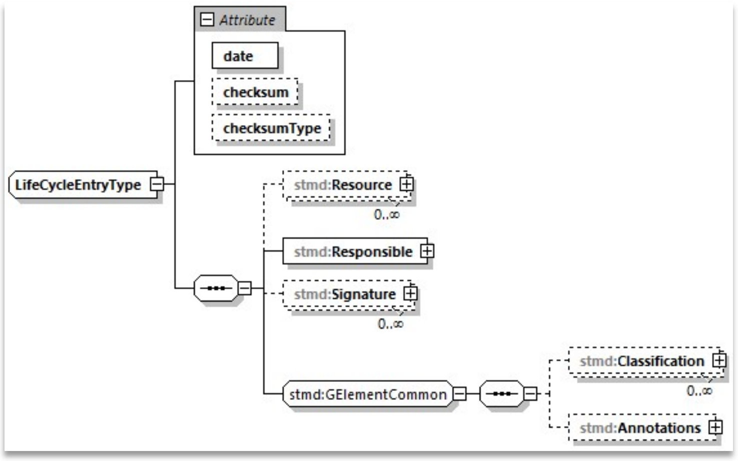


Figure 18: LifeCycleEntryType structure and attributes

The LifeCycleEntryType is structured by subordinated elements as described in the table below.

Element name	
LifeCycleEntryType	
Sub element name	Optional/mandatory
Resource	optional
Responsible	mandatory
Signature	optional
Classification	optional
Annotations	optional

Table 23: Sub elements of LifeCycleEntryType

The LifeCycleEntryType is associated with the following attributes as described in the table below.

Element type name		
LifeCycleEntryType		
Elements node XPath (if available)		
Attribute name	Optional/Mandatory	Attribute description
date	mandatory	Timestamp when life cycle entry was assigned. Note that the time stamp data

		type makes time zone information mandatory, so that a full ordering of times is possible.
checksum	optional	This attribute gives the checksum over the phase/step information stored in the enclosing phase/step element, calculated according to the STMD specification. This attribute is optional if the life cycle stage is not Approved or Archived, but becomes required if the life cycle stage is Approved or Archived. Optionally, digital signatures over this checksum can be provided using Signature elements in the enclosing life cycle entry element. The checksum is calculated using the algorithm indicated by the checksumType attribute.
checksumType	optional	This attribute gives the algorithm for the calculation of the checksum attribute. MUST be SHA3-256 for now, indicating a SHA3 256bit secure hash algorithm, as specified in FIPS 202. In the future other checksum algorithms might be supported.

Table 24: LifeCycleEntryType attributes

5.5.1 Resource

For details of the Resource structure and attributes see chapt 5.8 ResourceType.

5.5.2 Responsible

For the details of Responsible structure and attributes see chapt. 5.6 ResponsibleType.

5.5.3 Signature

For the details of Signature structure and attributes see chapt. 5.7 SignatureType.

5.5.4 Classification

For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.5.5 Annotations

For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.6 ResponsibleType

This element gives information on the responsible entity for a given step.

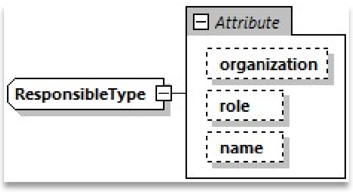


Figure 19: ResponsibleType structure

The ResponsibleType element is not structured by subordinated elements.

The ResponsibleType is associated with the following attributes as described in the table below.

Element name		
ReponsibleType		
Elements node XPath (if available)		
Attribute name	Optional/Mandatory	Attribute description
organization	optional	This attribute gives the organization that is responsible for a given step.
role	optional	This attribute gives the role of the person that is responsible for a given step.
name	optional	This attribute gives the name of the person that is responsible for a given step.

Table 25: ResponsibleType attributes

5.7 SignatureType

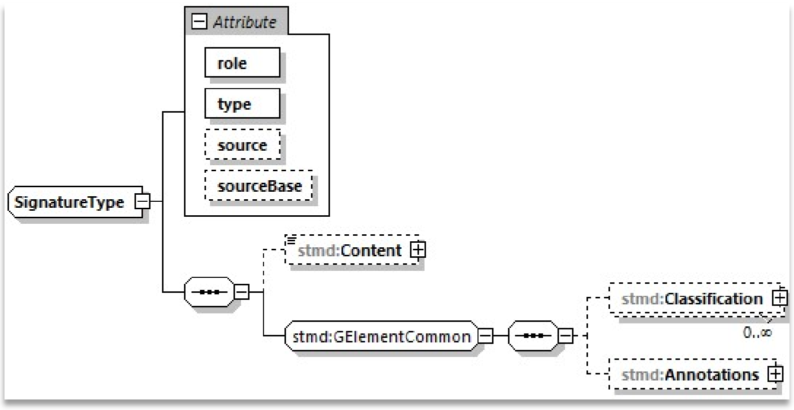


Figure 20: SignatureType structure

The SignatureType is structured by the subordinated elements as described in the table below.

Element name	
SignatureType	
Sub element name	**Optional/mandatory**
Content	optional
Classification	optional
Annotations	optional

Table 26: Sub elements of SignatureType

The SignatureType is associated with the following attributes as described in the table below.

Element name		
SignatureType		
Elements node XPath (if available)		
Attribute name	Optional/Mandatory	Attribute description
role	mandatory	This mandatory attribute specifies the role this signature has in the overall process. It indicates whether the digital signature is intended to just convey the authenticity of the information, or whether a claim for suitability of the information for certain purposes is made. In the latter case, the digital signature format should include detailed information about what suitability claims are being made.
type	mandatory	This mandatory attribute specifies the MIME type of the resource signature, which does not have a default value. If no specific MIME type can be indicated, then the type application/octet-stream is to be used.
source	optional	This attribute indicates the source of the resource signature as a URI (cf. RFC 3986). For purposes of the resolution of relative URIs the base URI is the URI of the STMD, if the sourceBase attribute is not specified or is specified as STMD, and the URI of the referenced resource if the sourceBase attribute is specified as resource. This allows the specification of signature sources that reside inside the resource (e.g. an FMU) through relative URIs. For signatures that are located alongside the STMD, relative URIs without scheme and authority can and should be used to

Kommentiert [CF2]: Spaltenbreiten optimieren, so dass die Tabelle insgesamt nicht so hoch wird.
(Gilt genauso auch für andere Tabellen von der Art.)

		specify the signature sources. For signatures that are packaged inside an SSP that contains this STMD, this is mandatory (in this way, the STMD URIs remain valid after unpacking the SSP into the filesystem). If the source attribute is missing, the signature is provided inline as contents of the Content element, which must not be present otherwise.
SourceBase	optional	Defines the base the source URI is resolved against: If the attribute s missing or is specified as STMD, the source is resolved against the URI of the STMD, if the attribute is specified as resource the URI is resolved against the (resolved) URI of the resource source.

Table 27: SignatureType attributes

5.7.1 Content

For details of the Content structure and attributes see chapt. 5.9 ContentType

5.7.2 Classification

For the details of Classification structure and attributes see chapt. 5.10 Classification.

5.7.3 Annotations

For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.8 ResourceType

This element provides information on one resource that is related to the particular step and particle. Multiple (or no) resources may be present.

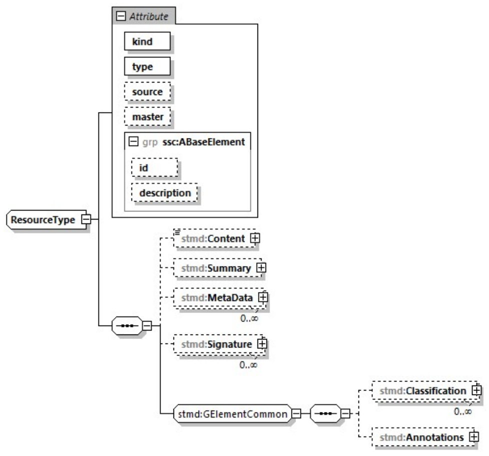


Table 28: ResourceType structure and attributes

The ResourceType element is structured by subordinated elements as described in the table below.

Element name	
ResourceType	
Sub element name	Optional/mandatory
Content	optional
Summary	optional
MetaData	optional
Signature	optional
Classification	optional
Annotations	optional

Table 29: Sub elements of ResourceType

The ResourceType is associated with the following attributes as described in the table below.

Element type name		
ResourceType		
Elements node XPath (if available)		
Attribute name	Optional/mandatory	Attribute description
kind	mandatory	This attribute indicates the kind of resource that is referenced, i.e. what role it plays in relation to the particle being described.
type	mandatory	This mandatory attribute specifies the MIME type of the resource, which does not have a default value. If no specific MIME type can be indicated, then the type application/octet-stream is to be used.
source	optional	This attribute indicates the source of the resource as a URI (cf. RFC 3986). For purposes of the resolution of relative URIs the base URI is the URI of the STMD. Therefore for resources that are located alongside the STMD, relative URIs without scheme and authority can and should be used to specify the component sources. For resources that are packaged inside an SSP that contains this STMD, this is mandatory (in this way, the STMD URIs remain valid after unpacking the SSP into the file system). If the source attribute is missing, the resource is provided inline as contents of the

		Content element, which must not be present otherwise.
master	optional	This attribute, if present, indicates the original, canonical master source for the resource. If it is present, it indicates that the content provided via source attribute and/or Content element is only a copy of the original, canonical data, and this attributes provides the URI reference to that original canonical master data.
id	optional	This attribute gives the model element a file-wide unique id which can be referenced from other elements or via URI fragment identifier
description	optional	This attribute gives a human readable longer description of the model element, which can be shown to the user where appropriate.

Table 30: ResourceType attributes

5.8.1 Content

For details of the Content element see chapt. 5.9 ContentType

5.8.2 Summary

This element provides an optional summary of the resource being referenced. The summary information is intended for human consumption to get an overview of the resource content without looking at the content itself. The summary content can be provided inline through the Content element, or it can be provided through the source URI attribute.

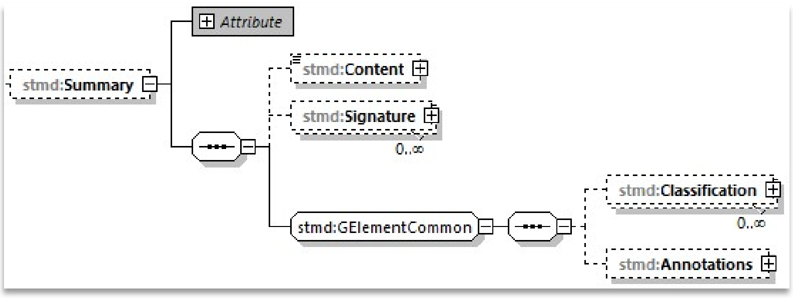


Table 31: Summary element structure

The Summary element is structured by the subordinated elements as described in the table below.

Element name	
Summary	
Sub element name	Optional/mandatory

Content	optional
Signature	optional
Classification	optional
Annotation	optional

Table 32: Summary element structure

The Summary element is associated with the following attributes as described in the table below.

Element name		
Summary		
Elements node XPath (if available)		
//element(*,stmd:ResourceType)/stmd:Summary		
Attribute name	Optional/mandatory	Attribute description
type	mandatory	This mandatory attribute specifies the MIME type of the resource summary, which does not have a default value. If no specific MIME type can be indicated, then the type application/octet-stream is to be used. If markdown content is used, then the type text/markdown shall be used.
source	optional	This attribute indicates the source of the resource summary as a URI (cf. RFC 3986). For purposes of the resolution of relative URIs the base URI is the URI of the STMD, if the sourceBase attribute is not specified or is specified as STMD, and the URI of the referenced resource if the sourceBase attribute is specified as resource. This allows the specification of summary sources that reside inside the resource (e.g. an FMU) through relative URIs. For summaries that are located alongside the STMD, relative URIs without scheme and authority can and should be used to specify the summary sources. For summaries that are packaged inside an SSP that contains this STMD, this is mandatory (in this way, the STMD URIs remain valid after unpacking the SSP into the filesystem). If the source attribute is missing, the summary is provided inline as contents of the Content element, which must not be resent otherwise.
SourceBase	optional	Defines the base the source URI is resolved against: If the attribute is

		missing or is specified as STMD, the source is resolved against the URI of the STMD, if the attribute is specified as resource the URI is resolved against the (resolved) URI of the resource source.

Table 33: Summary element attribute

5.8.2.1 Content

For the details of Content structure and attributes see chapt. 5.9 ContentType.

5.8.2.2 Signature

For the details of Singature structure and attributes see chapt. 5.7 SignatureType.

5.8.2.3 Classification

For the details of Classification structure and attributes see chapt. 5.10 Classification

5.8.2.4 Annotations

For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.8.3 MetaData

This element can specify additional meta data for the given resource. Multiple (or no) MetaData elements may be present.

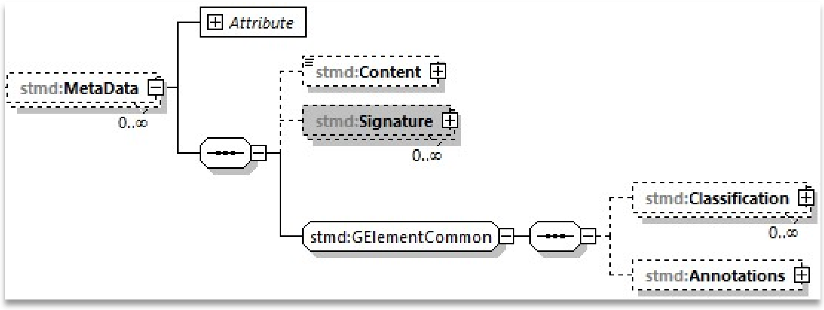


Figure 21: MetaData structure

The MetaData element is structured by subordinated elements as described in the table below.

Element name	
MetaData	
Sub element name	**Optional/mandatory**
Content	optional
Signature	optional
Classification	optional
Annotations	optional

Figure 22: MetaData sub elements

The MetaData element is associated with the following attributes as described in the table below.

Element name		
Metadata		
Elements node XPath (if available)		
//element(*,stmd:ResourceType)/stmd:MetaData		
Attribute name	Optional/Mandatory	Attribute description
kind	mandatory	This attribute indicates the kind of resource meta data that is referenced, i.e. what role it plays in relation to the resource being described.
type	mandatory	This mandatory attribute specifies the MIME type of the resource meta data, which does not have a default value. If no specific MIME type can be indicated, then the type application/octet-stream is to be used.
source	optional	This attribute indicates the source of the resource meta data as a URI (cf. RFC 3986). For purposes of the resolution of relative URIs the base URI is the URI of the STMD, if the sourceBase attribute is ot specified or is specified as STMD, and the URI of the referenced resource if the sourceBase attribute is specified as resource. This allows the specification of meta data sources that reside inside the resource (e.g. an FMU) through relative URIs. For meta data that are located alongside the STMD, relative URIs without scheme and authority can and should be used to specify the meta data sources. For meta data that are packaged inside an SSP that contains this STMD, this is mandatory (in this way, the STMD URIs remain valid after unpacking the SSP into the file system). If the source attribute is missing, the meta data is provided inline as contents of the Content element, which must not be present otherwise.
SourceBase	optional	Defines the base the source URI is resolved against: If the attribute is missing or is specified as STMD, the source is resolved against the URI of the STMD, if the attribute is specified as resource the URI is resolved against the (resolved) URI of the resource source.

Figure 23: MetaData sttributes

5.8.3.1 Content
For details of the Content structure and attributes see chapt. 5.9 ContentType

5.8.3.2 Signature
For the details of Signature structure and attributes see chapt. 5.7 SignatureType.

5.8.3.3 Classification
For the details of Classification structure and attributes see chapt.5.10 Classification

5.8.3.4 Annotations
For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.8.4 Signature
For the details of Classification structure and attributes see chapt. 5.7 SignatureType.

5.8.5 Classification
For the details of Classification structure and attributes see chapt.5.10 Classification

5.8.6 Annotations
For the details of Annotations structure and attributes see chapt. 5.11 Annotations.

5.9 ContentType

This optional element can contain inlined content of an entity. If it is present, then the attribute source of the enclosing element must not be present.

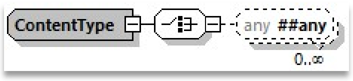


Figure 24; ContentType structure

The ContentType element is structured by subordinated elements as described in the table below.

Element name	
ContentType	
Sub element name	Optional/mandatory
##any	optional

Table 34: Sub elements of ContentType

The ContentType is not associated with any attributes.

5.9.1 ##any
The ContentType may contain XML Elements of any kind, i.e. the STMD Schema provides the possibility and capability to code any kind of information regardless of what the STMD specifies. This mean the name, structure and attributes of XML elements enclosed by a contentType element is completely free.

5.10 Classification

This element, which can occur multiple times, provides a set of classifications of an STMD modeling element, provided as Keyword Value Pairs (KWP), the meaning of which is interpreted according to the name of the classification provided in the name attribute. This approach can be used, for example, to provide searchable keywords for content, or to assign and track quality or validation level requirements across the STMD process, or to maintain variant or other classification content across the process.

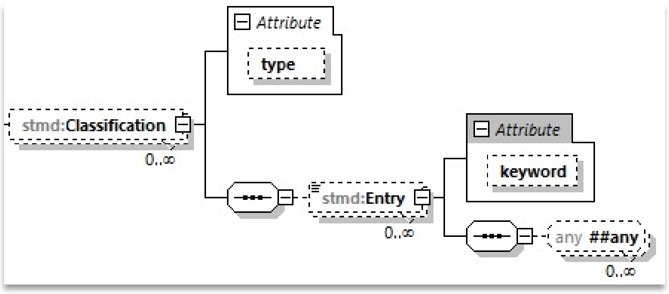


Figure 25: Classification structure and attributes **(To be replaced)**

The Classification element is structured by the subordinated elements as described in the table below.

Element name	
Classification	
Sub element name	Optional/mandatory
Entry	optional

Figure 26: Sub elements of Classification

The Annotation element is associated with the following attributes as described in the table below.

Element name		
Classification		
Elements node XPath (if available)		
//element(*,stmd:ResourceType)/stmd:Annotations/ssc:Annotation		
Attribute name	Optional/Mandatory	Attribute description
type	mandatory	This attribute provides the name of the type of classification being provided. The name should be unique across the Classification elements of the immediately enclosing element. In order to ensure uniqueness all types should be identified with reverse domain name notation (cf. Java package names or Apple UTIs) of a domain that is controlled by the entity defining the semantics and content of the classification.

Figure 27: Classification attributes

5.10.1 ClassificationEntry

The ClassificationEntry element is structured by the subordinated elements as described in the table below.

Element name	
ClassificationEntry	
Sub element name	Optional/mandatory
##any	optional

Figure 28: Sub elements of ClassificationEntry

The ClassificationEntry element is associated with the following attributes as described in the table below.

Element name		
ClassificationEntry		
Elements node XPath (if available)		
//element(*,stmd:ResourceType)/stmd:Classification/stmd:Entry		
Attribute name	Optional/Mandatory	Attribute description
keyword	optional	This attribute gives the keyword for the classification entry (i.e. keyword value pair). It is left undefined whether this must be unique across the entries of the Classification element, or whether repeated entries are allowed. This will depend on the definition of the classification.

Figure 29: ClassificationEntry attributes

5.10.1.1 ##any

The Annotation element may contain XML Elements of any kind, i.e. the STMD Schema provides the possibility and capability to code any kind of information regardless of what the STMD specifies. This means, the name, structure and attributes of XML elements enclosed by a contentType element ~~is~~ are completely free.

Kommentiert [CF3]: Dieselben Korrekturen per "search & replace" auch an andere Stellen übertragen.

5.11 Annotations

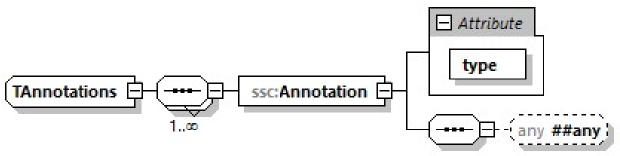


Figure 30: Annotations structure and attributes

The Annotations element is structured by the subordinated elements as described in the table below.

Element name	
Annotations	
Sub element name	**Optional/mandatory**
Annotation	mandatory

Figure 31: Sub elements of Annotations

The Annotations element is not associated with any attributes.

5.11.1 Annotation

The Annotation element is associated with the following attributes as described in the table below.

Element name		
Annotation		
Elements node XPath (if available)		
//element(*,stmd:ResourceType)/stmd:Annotations/ssc:Annotation		
Attribute name	**Optional/Mandatory**	**Attribute description**
type	mandatory	The unique name of the type of the annotation. In order to ensure uniqueness all types should be identified with reverse domain name notation (cf. Java package names or Apple UTIs) of a domain that is controlled by the entity defining the semantics and content of the annotation. For vendor-specific annotations this would e.g. be a domain controlled by the tool vendor. For MAP-SSP defined annotations, this will be a domain under the org.modelica prefix.

Figure 32: Annotation attributes

5.11.1.1 ##any

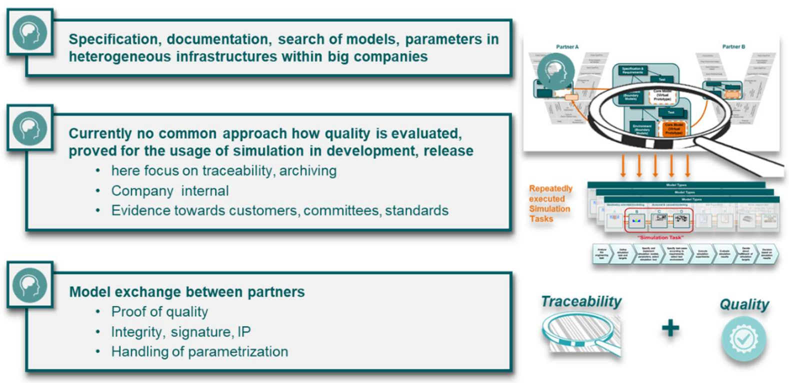
The Annotation element may contain XML Elements of any kind, i.e. the STMD Schema provides the possibility and capability to code any kind of information regardless of what the STMD specifies. This mean the name, structure and attributes of XML elements enclosed by a contentType element is completely free.

6 Working with STMD files

This Chapter is currently under rework/construction

6.1 Challenges in dealing with simulation tasks and simulation models

This chapter describes the key challenges in dealing with simulation models and parameters in the automotive industry (see Figure 33~~Figure 33~~). The following subchapters describe these challenges in details.


Specification, documentation, search of models, parameters in heterogeneous infrastructures within big companies

Currently no common approach how quality is evaluated, proved for the usage of simulation in development, release
- here focus on traceability, archiving
- Company internal
- Evidence towards customers, committees, standards

Model exchange between partners
- Proof of quality
- Integrity, signature, IP
- Handling of parametrization

Figure 33: Challenges in dealing with simulation tasks and simulation models

6.1.1 Specification, documentation and search of models and parameters

Internally within big companies, the specification and documentation of simulation models and simulation parameters and the search of models and parameters are essential business needs and big challenges for simulation data management and simulation task quality traceability.

Specification, documentation, search of models, parameters in heterogeneous infrastructures within big companies

Figure 34: Specification, documentation and search of models and parameters

6.1.1.1 Specification of simulation models and parameters

In all cases, when a simulation model is to be created, the following must be specified:

- what the model should serve for
- for which type of simulation the model should be created,
- which start and boundary conditions have to be considered,
- in which execution environment the simulation should be executed,
- which modeling-specific conditions are relevant,
- which ports and parameters should be used
- as well as many other aspects.

All of this is information that the creator of a model may need to know in order to create a model that meets all the requirements of the simulation task.

The challenge is to specify the requirements for a requested simulation model, so that the creator of a simulation model can efficiently provide a suitable model or parameter set for a simulation task.

The same challenge applies to defining simulation parameters and parameter sets.

6.1.1.2 Documentation of simulation models and parameter

Supplier and partners often supply simulation models as functional mock-ups (FMUs) and describe them by some metadata, for example

- the purpose of the FMU,
- the type of simulation for which the model was defined
- the starting and boundary conditions considered
- the intended execution environment of the FMU
- taken into account modeling-specific aspects
- possibly a list of ports and parameters

All this information can help to verify that the supplied simulation model meets the specified simulation model requirements with respect to the simulation.

The same challenge applies to defining simulation parameters.

6.1.1.3 Search and retrieval of models and parameters based on metadata

Retrieving simulation models and parameter sets is a core feature of simulation data management, and reusing the simulation models is an essential methodology to reduce preprocessing overhead. The effectiveness and efficiency of retrieving and reusing simulation and parameter sets is crucially dependent on search performance, which itself depends on the availability of searchable metadata with sufficiently high semantic quality.

Most of the appropriate metadata is the same information used to specify and document simulation models or FMUs.

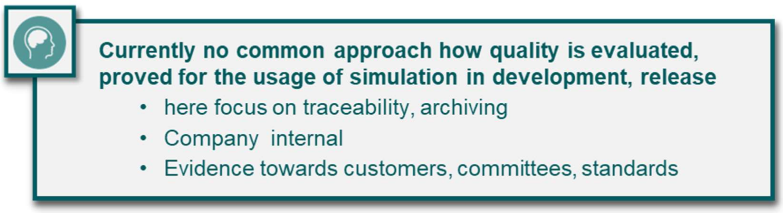
The same challenge applies to defining simulation parameters.

6.1.2 Quality traceability and proof of simulation tasks

The quality of simulation tasks or its results with respect to a higher-level engineering task essentially depends on the definition of the simulation task and the simulation data used. This includes, among other things, the simulation models and parameter sets used, the settings used during the execution of the simulation. Also of essential importance is the congruence of the simulation models and simulation settings with the requirements of the simulation task.

Great challenges are

- to document and trace the quality of a simulation task with all its associated simulation data and settings
- to proof and evaluate the quality of a simulation task with all its associated simulation data and settings



Currently no common approach how quality is evaluated, proved for the usage of simulation in development, release
- here focus on traceability, archiving
- Company internal
- Evidence towards customers, committees, standards

Figure 35: Quality evaluation for simulation tasks

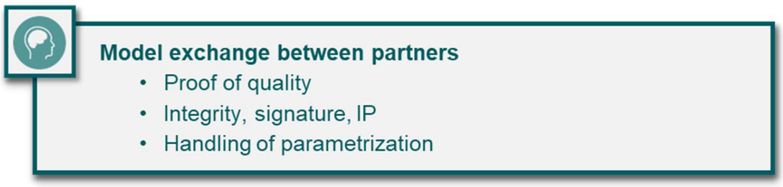
6.1.3 Exchange of simulation models and simulation parameters

Chapter 6.1.1 and 6.1.2 focus on internal business needs, although the needs described above may be due to external requirements. This chapter focuses on business requirements through the explicit exchange of simulation models or FMUs and simulation parameters.

Collaborative simulation scenarios may occur, among others, in the following constellations:

- Provision of simulation models and parameters for customers
- Request and receive simulation models and parameters from (sub-) suppliers
- Simulation collaboration with equivalent partners in engineering companies
- Provision of product-related technical information, e.g., for certification or homologation

In all these constellations, it is of considerable importance that the quality of simulation tasks or the associated simulation data and results is traceable and provable. This may also include aspects of data integrity and reliable digital signature of data, as well as intellectual property protection.



Model exchange between partners
- Proof of quality
- Integrity, signature, IP
- Handling of parametrization

Figure 36: Model Exchange between engineering partners

6.2 Functional and procedural environment

Kommentiert [CF4]: Wäre es ggf. motivierender, die Beschreibung von Prozess & Use Cases eher an den Anfang des ganzen Dokuments zu stellen?

6.2.1 Generic simulation-based decision process

The generic simulation-based decision process approach assumes that simulations tasks are defined and executed with the purpose of providing basic information for decisions. Figure 37~~Figure 37~~ outlines the generic simulation-based decision process.

Kommentiert [CF5]: Graphic needs to be updated.

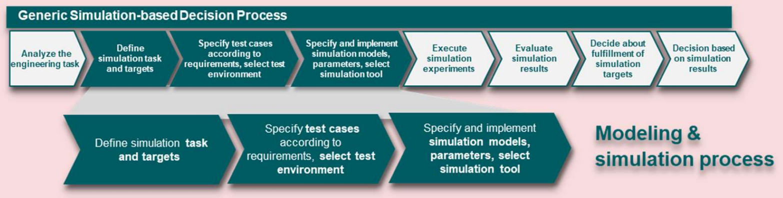

Generic Simulation-based Decision Process
Analyze the engineering task
Define simulation task and targets
Specify test cases according to requirements, select test environment
Specify and implement simulation models, parameters, select simulation tool
Execute simulation experiments
Evaluate simulation results
Decide about fulfillment of simulation targets
Decision based on simulation results
Define simulation task and targets
Specify test cases according to requirements, select test environment
Specify and implement simulation models, parameters, select simulation tool
Modeling & simulation process

Figure 37: Generic simulation based decision process

The green shaded process phases in Figure 37~~Figure 37~~entirely represent the core modeling & simulation process.

Usually a simulation engineer is responsible for the modelling & simulation process, but neither for the product or system design or nor for the project decisions that are made based on the evaluated simulation results. As soon as the requested simulation results are available, the simulation engineer closes the assigned simulation task.

To guarantee safe and reliable design and project decisions it is imperative that the decision bases on robust and reliable information respectively robust and reliable simulation results with a sufficiently proofed and traceable quality level.

6.2.2 Quality traceability

Essentially, guarantying quality traceability of simulation tasks means to document how the content and the quality of a simulation result has been achieved by executing the simulation task with additional consideration of when and where the execution steps have been executed.

Quality traceability is useful or necessary for several purposes, for example in case of an audit for quality proof or reasoning of project decisions. It is also useful or necessary for re-simulation of a simulation task with the same conditions (all sub summed as “ex-post” use cases) or for partially re-utilization of simulation task associated artifacts like simulation models, simulation parameters, etc. for similar simulation tasks (sub summed as “ex-ante”).

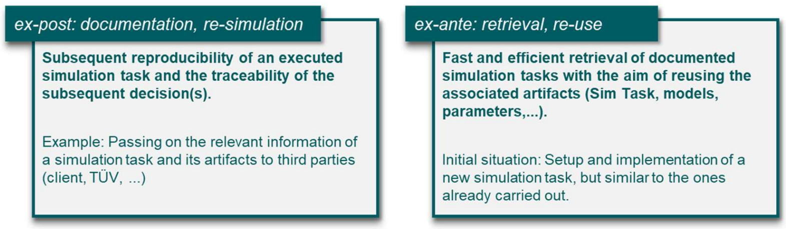

ex-post: documentation, re-simulation
Subsequent reproducibility of an executed simulation task and the traceability of the subsequent decision(s).
Example: Passing on the relevant information of a simulation task and its artifacts to third parties (client, TÜV, ...)
ex-ante: retrieval, re-use
Fast and efficient retrieval of documented simulation tasks with the aim of reusing the associated artifacts (Sim Task, models, parameters,...).
Initial situation: Setup and implementation of a new simulation task, but similar to the ones already carried out.

Figure 38: Ex-post und ex-ante use cases

6.2.3 Description of use cases

Notizen (ST= Simulation Task):

- ST aus Spezifikationssicht betrachten (am Beginn des Workflows)
- ST aus Dokumentationssicht betrachten (später im Workflow~~s~~)
- ST befüllen und ändern
  - Projektmanagementinformationen übernehmen (verknüpfen, ändern, entfernen)
  - Systemstruktur übernehmen (verknüpfen, ändern, entfernen)
  - Parameter übernehmen (verknüpfen, ändern, entfernen)
  - Simulationsinformationen übernehmen (verknüpfen, ändern, entfernen)
- Strukturzentrierte Sicht
- Prozesszentrierte Sicht

6.2.3.1 Use case: Create a simulation task

The creation of a new simulation task might occur in at least three different variants.

- Creation of a (neutral) simulation task
- Creation of a simulation task as a copy
- Creation of a simulation task from a template

The use case diagram in Figure 39~~Figure 39~~ shows the appropriated use cases with their base use case and their dependencies.

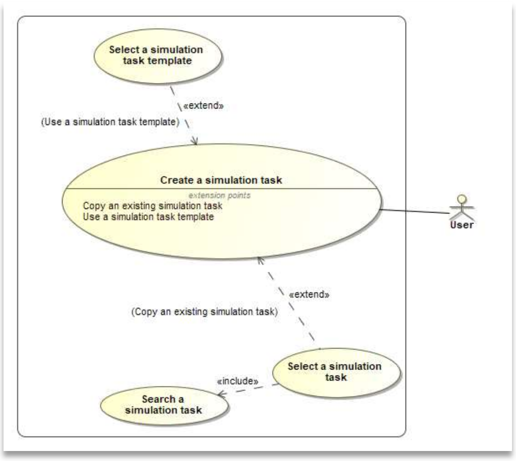

Select a simulation task template
«extend»
(Use a simulation task template)
Create a simulation task
extension points
Copy an existing simulation task
Use a simulation task template
User
«extend»
(Copy an existing simulation task)
Select a simulation task
«include»
Search a simulation task

Figure 39: Creating a simulation task

6.2.3.1.1 Primary use case: Create a simulation task

The use case creates a new simulation task that does not contain any data, probably except for some metadata, like e.g. creation date and time. This use case might be extended by using a template for new simulation task, i.e. by selecting an existing template that already contains some template specific payload data (see chapter 6.2.3.1.2) or the use case might be extended by creating the new simulation task as a copy of an existing simulation task (see chapter 6.2.3.1.3).

Actually, this use case implicitly also uses a template but this template is an empty template without any application specific entries.

6.2.3.1.2 Primary use case: Create a simulation task as a copy

Creating a simulation task as a copy of an existing simulation task is quite similar to the use case Create a simulation task extended by an additional step to select the existing simulation task to be used as the base for the copy. This is modeled in Figure 39~~Figure 39~~ by the <<extends>> relationship and the extension point Copy an existing simulation task template in the use case Create a simulation task. To select a copy base for the simulation task, one might use search functionality.

Note: The use case creates a new simulation task with the initial stage of the simulation task workflow. This is to be clearly distinguished form creating a branch of an existing simulation task. A new branch of an existing simulation task inherits the stage of the base simulation task at the moment of creation.

6.2.3.1.3 Use case: Create a simulation task from a template

Creating a simulation task from a template is quite similar to the use case *Create a simulation task,* extended by an additional step to select the template to be used. This is modeled in Figure 39~~Figure 39~~ by the <<*extends*>> *relationship* and the extension point *Use a simulation task template* in the use case *Create a simulation task*.

6.2.3.2 Create a branch of an existing simulation task

tbd

6.2.3.3 Edit a simulation task

TBD

6.2.3.4 Search a simulation task

Searching a simulation task might be needed either explicitly or be involved implicitly by other use cases, e.g. to search an existing simulation task a copy base for creating a simulation task (see 6.2.3.1.2)

The search for a simulation task might occur in at least three different variants.

- Searching for a simulation task by navigation through a repository
- Search for a simulation task by ID or name
- Searching for a simulation task by querying some additional metadata

The use case diagram in Figure 40~~Figure 40~~ shows the appropriated use cases with their dependencies.

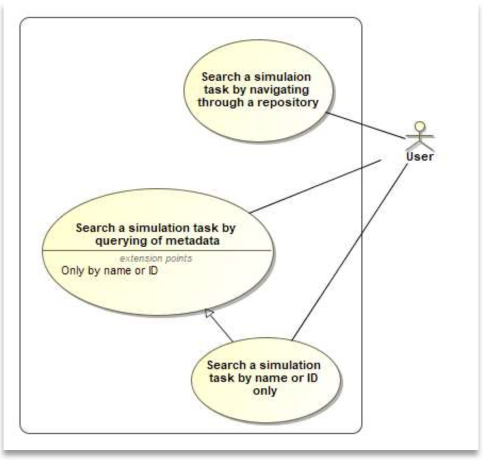

Search a simulaion task by navigating through a repository
User
Search a simulation task by querying of metadata
extension points
Only by name or ID
Search a simulation task by name or ID only

Figure 40: Searching a simulation task

6.2.3.4.1 Search a simulation task by navigation through a repository

This use case assumes that simulation tasks are stored in a repository resp. are managed with data management feasibilities. Navigating through a repository means that the user gets an interactive view into the repository and can navigate at least top-down and bottom-up through the repository structure.

6.2.3.4.2 Search a simulation task by querying of metadata

This use case assumes that simulation tasks are searchable by some predefined metadata fields. To search a simulation task by query also requires a kind of query builder that facilitates to build a search logic that combines several metadata fields. The query would respond with a list of one or more search results from which a user can pick a specific simulation task.

6.2.3.4.3 Search a simulation task by name or ID only

The search for a simulation task by ID or name is actually also a query but simplified to a narrow search that only searches in one of two metadata fields.

7 Rules and recommendations

The following list of global rules are considered valid for the entire STMD definition.

Rule 1: For ***all*** filled LifeCycleInformation entries (i.e. Drafted, Defined, Validated, Approved; Archived or Retracted) on phase level ***shall be*** considered valid for all subordinated steps of that phase.

Rule 2: A filled LifecycleInformation entry of ***Validated* or Approved *should*** carry a signature and a checksum.

Rule 3: As soon as a certain ***phase is not started***, i.e. no phase specific information is documented at all, the respective phase XML element ***should not*** be instantiated.

Rule 4: As soon as a certain ***phase is started***, i.e. at least one phase or subordinated step specific information is already documented, all subordinated step XML elements of the phase ***should*** be instantiated.

Rule 5: …

8 Requirements for tool based functionalities

8.1 Open system structure package

Tbd

8.2 Read a single STMD file

Tbd

8.3 Load referenced glue particle

Tbd

8.4 Process read/write access rights

Tbd

8.5 Edit an STMD file

(Inputs, Procedures, Outputs, Rationales) Tbd

8.6 Add and edit classification

Tbd

8.7 Add and edit annotations

Tbd

8.8 Create LifeCycleInformation

tbd

8.9 Validate an STMD file

Tbd

8.10 Save an STMD file

Tbd

8.11 Create references to other glue particle

Tbd

8.12 Create System structure package

tbd

8.13 Validate all checksums of an STMD file

Tbd

8.14 Check and validate external references

tbd

8.15 Evaluate consistency of information

Tbd

8.16 Visualize content of an STMD file

Tbd

8.17 Compare two STMD file

Tbd

8.18 Markup differences of two STMD files

Tbd

8.19 Create a branch of an STMD file

Tbd

8.20 Sign glue particle

(Create signature) tbd

8.21 Create Checksum

tbd

9 Instantiation example: Simulation of a DC Motor

Description of scenarios including the description of instantiation examples (referenced STMD XML files)

10 Glossary

Embedded or externally defined and referenced directory of terms, abbreviations and definitions

11 Glossary