OETCS/WP7/D7.3

openETCS

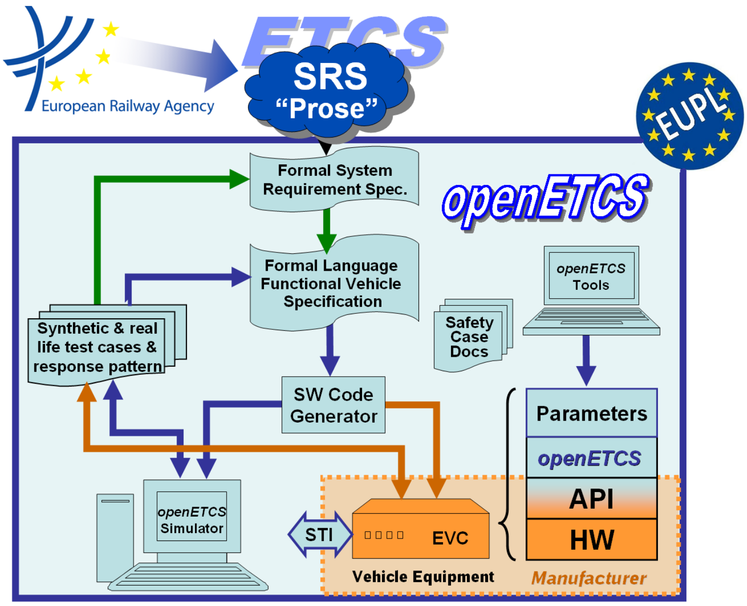
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 7: “Toolchain”

Toolchain Qualification Process Description

Cecile Braunstein, Jan Peleska, Stefan Rieger and Izaskun De La Torre

January 2014



Funded by:

This page is intentionally left blank

Work-Package 7: "Toolchain"

OETCS/WP7/D7.3
January 2014

Toolchain Qualification Process Description

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Cécile Braunstein (University Bremen)	signature Michael Jastram (Formal Mind)	signature Marielle Petit-Doche (SQS)	signature Klaus-Rüdiger Hase (DB Netz)

Cecile Braunstein and Jan Peleska

University Bremen

Stefan Rieger

TWT GmbH Science & Innovation
Ernsthaldenstraße 17
70565 Stuttgart
Germany

Izaskun De La Torre

SQS

Qualification process description

Prepared for openETCS@ITEA2 Project

Abstract: This document presents different ideas of a toolchain qualification. It describes a process for the openETCS toolchain qualification.

Table of Contents

Document Information

Document information	
Work Package	WP7
Deliverable ID or doc. ref.	D7.3
Document title	Toolchain Qualification Process Description
Document version	01.11
Document authors (org.)	Cécile Braunstein and Jan Peleska (Uni.Bremen)

Review information	
Last version reviewed	01.11
Main reviewers	Marielle Petit-Doche, Michael JastramJ, Jaime Paniagua

Approbation			
	Name	Role	Date
Written by	Cécile Braunstein	WP7-T7.3 Sub-Task Leader	12.01.2014
Approved by			

Document evolution			
Version	Date	Author(s)	Justification
00.00	12.01.2014	C. Braunstein	Document creation
01.00	19.01.2014	C. Braunstein	Jan Peleska suggestions
01.01	28.01.2014	M. Jastram	Review
01.02	28.01.2014	C. Braunstein	Change from MPD, MJ JP suggestions
01.03	08.05.2014	S. Rieger	Provided first draft of a tool qualification example
01.04	08.05.2014	C. Braunstein	Provided another tool qualification example
01.05	27.05.2014	S. Rieger	Provided initial list of tool qualification scenarios
01.06	02.07.2014	S. Rieger	Added preliminary tool chain process visualisations
01.07	14.07.2014	S. Rieger	Model-based tool qualification, still incomplete
01.08	24.07	Izaskun De la Torre	Incremental Tool chain qualif
01.08	30.07.2014	C. Braunstein	Organize chapter 2
01.09	04.08.2014	S. Rieger	Refined Chapter 2
01.10	08.08.2014	C. Braunstein	Insert Izaskun De La Torre remarks
01.11	12.08.2014	S. Rieger	Took into account Izaskun De La Torre's remarks
01.12	02.10.2014	C. Braunstein	Integrate WP2 review

1 Introduction to Toolchain Qualification

1.1 Tool Qualification

The CENELEC EN 50128 standard [12] defines the tool qualification as follows:
"The objective is to provide evidence that potential failures of tools do not adversely affect the integrated tool-set output in a safety related manner that is undetected by technical and/or organizational measures outside the tool. To this end, software tools are categorized into three classes namely, T1, T2 & T3 respectively."

We recall here the different class definitions:

- Tool class T1: No generated output can be used directly or indirectly to the safetey critical executable code;
- Tool class T2: Verification tools, e.g. that can not introduce errors to the safety critical executable code but those tools may fail to detect errors or defects;
- Tool class T3: Generated output directly or indirectly as part of the safety critical executable code.

The deliverable D2.2 [9] summarizes the requirements for the tool needed by the different tool classes. The report highlights that the effort differs depending on the tool class. Furthermore, for the most critical class T3, the evidence should be provided that the output is conform to the specification or that *any failure in the output are detected*.

The standard defined how to classify each tool individually (see [7, 8] as an example). But dealing with a tool chain, integrated within a tool platform, implies extra effort to ensure that the tool integration does not introduce new errors. For example mechanism such as artifacts versioning, time-stamping operations, etc ... should also be considered when qualifying the tool chain. This increased the number of tools to consider during the qualification process.

The effort of qualification depends on the number of tools under consideration, the tool classes and the tool error detection capabilities. To reduce the cost of the toolchain qualification and regarding the fact that our development imply regular releases, a systematic toolchain analysis approach has to be defined.

Table 1 exemplarily lists the tool classes for some of the tools employed in the openETCS project. Automatic code generator for SCADE model such as KCG or the Bitwalker tool have a direct effect on the generated code whereas ProR and Git do not. The classification for ProR would be different if it would be used for formal requirements that then could be automatically translated to a model (this would yield a T3-classification). The verification tools RT-Tester and CPN Tools are not in the "direct" chain from requirements to code but they may fail to detect errors in the software or model.

Tool	Purpose	Tool Class
Papyrus Editor	Definition of the model architecture	T1
Papyrus SysML checker	Check SysML conformity of the model	T2
SCADE Editor	Low-level modelling and code generation	T1
SCADE Code Generator	Code generation	T3
ProR	Requirements management	T1
Bitwalker	Generation of data structures for modelling	T3
Git	Versioning & Traceability	T1
RTTester	Model-based testing	T2
CPN Tools	Model checking and test case generation	T2

Table 1. Classification of some tools in the openETCS tool chain

1.2 Toolchain Qualification State of the Art

Some recent works have been done in the field of toolchain qualification from a variety of projects. The next section summarizes the most significant ones.

1.2.1 Slotosh and al. (project RECOMP)

[10] describes a model-based approach to tool qualification to comply with DO-330 and integration into the Eclipse development environment. The authors claim that the benefits of their method are the following:

Clarity: remove ambiguities;

Re-usability and Transparency: check for reuse in different toolchain;

Completeness: the model covers all parts of the development process and tis traceable;

Automation: Some part of the process may be automated.

Their method is explained in detail in [11]: the toolchain analysis is based on a domain specific toolchain model they have defined. This model is used to represent the toolchain structure as well as the tool confidence. Their goal is to deduce the tool confidence level and to expose specific qualification requirements. Furthermore, their idea is not only to check tool by tool but they follow a more holistic approach that makes use of rearrangement and/or the extension of the toolchain to avoid the certification of all tools. This allows them to reduce the qualification effort by focusing only on the critical tools and making use of already available information. Moreover, checks with respect to inconsistencies, such as missing descriptions, unused artifacts, etc., may be automatically executed on the toolchain model. Finally, automated document generation is addressed.

In [13] the application of tools and methods to an industrial use case to determine the potential errors in the tool-chain is described.

1.2.2 Asplund and al. (projects iFEST, MBAT)

The authors investigate the question if there exists part of the environment related to tool integration that may fall outside the tool qualification defined by the a norm (ISO 26262 here [2]). And if so, how tool integration is affected by ensuring functional safety. One conclusion is that the tool integration may lead to an increase of the qualification effort.

They also state that the standards (EN 50128, DO-178C and ISO26262) are not sufficient to check safety of a toolchain, but some part of a toolchain may be taken into account to mitigate the qualification effort. They highlight 9 safety issues caused by tool integration that also allow to be more exact when identifying software that have to be qualified for certification purpose.

They advocate to use take a “system approach” to deal with the qualification of tool integration within a toolchain. We should not think about individual tools anymore. Their system approach follows these steps:

1. Pre-qualification of development tools (requirements tools, design tools ...): provided by the vendors.
2. Pre-qualification at the tool-chain level: based on step 1 and reference work-flows; decomposition of higher level (project-wide) safety goals on tool level
3. Qualification at the toolchain level: check whether assumptions in step are fulfilled (use cases, environment, process) by the actual toolchain to be deployed.
4. Qualification at the tool level: based on the actual environment when deploying the toolchain.

This approach leads them to separate the parts required for software tool qualification and to identify safety issues related to tool integration.

In [1], they explore the step 2): identifying the required safety goals due to tool integration and obtaining a description of a reference work-flow and tool-chain with annotations regarding the mitigating effort. They propos to use the TIL language, a domain specific language for toolchain models. The model of the tool chain is used to perform a risk analysis and to annotate parts that need mitigating effort for the safety issues due to tool integration.

1.2.3 Biehl and al. (projects CESAR, iFEST, MBAT)

Biehl proposed a Domain Specific Language named TIL for Generating Tool Integration Solutions [6]. A toolchain is described in terms of a number of “Tool Adapters” and the relation between them.

- Tool Adapters: expose data and functionality of a tool
- Channels
  - ControlChannel describes service calls
  - DataChannel describes data exchanges
  - TraceChannel describes creation of a trace links
- Sequencer: describes sequential control flow (sequence of services)
- User: describes and limit the possible interaction

- Repository: provides storage and version management of tool data

This DSL allows early analysis of the toolchain. It may generate part of tool adapter code based on the source and target meta-model.

More recently, Biehl and al. define a standard language for modeling development processes as defined by OMG 2008. The language has been used in [5, 4] together with the TIL language to tailor a toolchain following a process model. The goal is to be able to model both the development process and the set of tools used. A process is defined as follows:

- Process: several Activities
- Activity: set of linked Tasks, WorkProducts, Roles
- A Role can perform a Task
- A WorkProduct can be managed by a Tool
- A Task can use a Tool

Using together the process development language and the toolchain language, in [4], the authors measure the alignment of a toolchain with a product development process. The method proceeds as follows:

1. Inputs:
   - formalized description of the toolchain design
   - description of the process including the set of tools and their capabilities
2. Initial verification graph
3. Automatic mapping links to the verification graph (acc. to mapping rules)
4. Apply alignment rule on the verification graph
5. Apply metrics to determine the degree of alignment between the tool-chain and the process

The metrics and the misalignment list provide feedback to refine the tool-chain design.

2 OpenETCS Toolchain Qualification Process

2.1 OpenETCS Tool Chain Characteristics

All the methods mentioned above start with a complete definition of the toolchain. In OpenETCS, the development of the toolchain follows an *agile* approach. Hence, for each (major) release we have to deal with an incomplete tool chain. In addition to the methods of the previous section, we need a qualification process that can adapt to the development speed, deal with an incomplete toolchain and can re-use qualification information.

Moreover, as stated by Asplund et al., the toolchain itself may provide some mechanisms that may reduce the tool qualification effort [1, 2]. They are described as a set of safety goals that the tool chain should ensure. In our context, most of the tool integration effort is made by integrating tools into a tool platform. According to Asplund et al., the tool platform should ensure the following safety-goals that will avoid some extra tool qualification:

- Coherent time stamp information: common time stamps on development artifacts.
- Notification: the user should be notified when artifacts changed.
- Data integrity: avoid use of obsolete artifacts, the data used reflects the current state.
- Data mining: all data used by safety analysis should be available and be verifiable.

These goals will be included in our tool integration qualification, they will be extra requirements for each tool qualification. Moreover, a set of specific requirements has been described in the deliverable D2.6 [3]. These should also been included in the tool's requirements.

In the following sections we define the tool chain qualification process by first giving an overview of the process (section 2.2), then getting into details for individual tool qualification (section 2.3), we then focus on the incremental development of the tool chain (section 2.4) and finally provide some example on how this process can be applied to our tool chain implementation (section 2.5).

2.2 Qualification Process Overview

Figure 1 defines the qualification process.

The steps of the process are defined as follows:

Feature Analysis

The tool chain is defined as feature or activity (e.g. "Requirements Engineering", "Documentation", "Modelling") . Each tool of the tool chain is then categorised according to its activities. Inside a feature a work-flow is defined.

This phase also defines the interfaces of each tool within a feature and the artifacts manipulated. An artifact is defined by a name and a format.

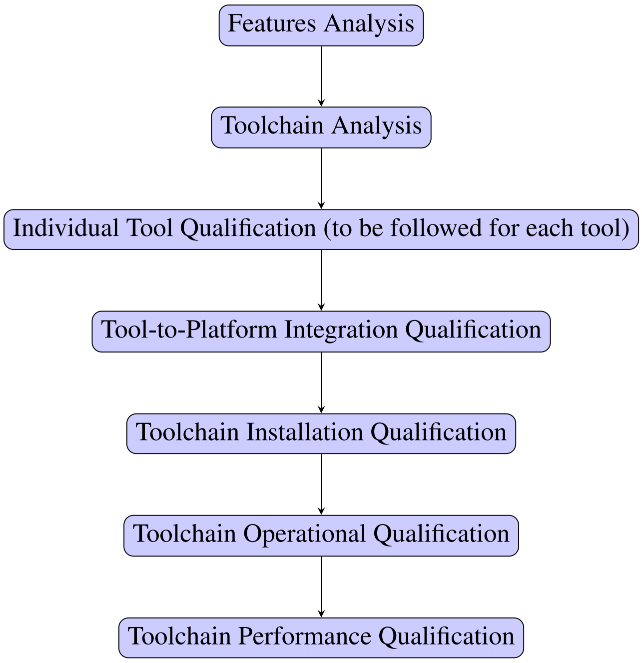


Figure 1. Proposal for the openETCS qualification process

Tool chain Analysis

The work-flow between the tool chain features is defined. The interfaces between the features are defined at this phase as well as the artifacts exchange between the different features.

Individual Tool Qualification (to be followed for each tool)

See section 2.3

Tool-to-Platform Integration Qualification

This step ensures that the tool is correctly integrated within the tool chain. After all individual tools are qualified, their integration into the tool chain is planned. As part of the integration planning, the team will analyse the dependencies of the different tools to minimise the effects of unavoidable dependencies.

Moreover, for each artifact used or produced by the tool, evidence should be provided that the safety-goals of section 2.1 are satisfied. Whenever an artifact is modified, a new time stamp has to be added and notification should be triggered. Whenever an artifact is used a check of its obsolescence should be performed.

Tool Chain Installation Qualification

Installation should verify that the toolchain is properly installed. It does not verify that the tool chain conforms to the functional and performance specification. This is done later in the operational qualification phase. The goal of this step is to verify correct software installation and to document all computer hardware, software and configuration settings as the initial baseline configuration.

Operational qualification

Operation qualification should demonstrate that the tool chain will function according to its operational specification in the selected environment. The tests should be performed to verify that the toolchain meets the specifications, requirements in the specific environment.

Performance qualification

Performance qualification should demonstrate that the toolchain consistently performs according to the specifications defined by the openETCS project, and is appropriate for the intended use. Important for consistent openETCS toolchain performance is regular preventive maintenance, making changes to the toolchain in a controlled manner and regular testing.

The toolchain should be well maintained to ensure proper ongoing performance. Procedures should be in place for regular preventive maintenance of it to detect and fix problems before they can have a negative impact.

2.3 Individual Tool Qualification

The qualification of a tool chain implies that each tool should be qualified. Figure 2 proposes a tool qualification process that should be performed for each tool. This section explains the steps for the tool qualification whenever a new tool is added. Next section will deal with the special cases where tools are updated and when a first qualification has already been performed.

2.3.1 Define Process & Usage Context

The process starts by defining the tool purposes and the usage context of tool. More precisely, the formats restrictions of the input and output artifacts of the tool should be identified, the dependency with our tools should be explicitly defined and the feature analysis should be consolidate. This can be best achieved in a model-based manner. For details on that topic please refer to Section 2.3.4.

2.3.2 Tool Classification

The class of the tool should be set according to the EN50128 definition (see section 1.1). An evidence can be expressed using the feature and the tool chain analysis.

These two first steps may be seen as pre-qualification steps that guides the qualification process.

2.3.3 Scenario-based Qualification of Individual Tools

Qualifying a toolchain always requires the qualification of the individual tools it is comprised of. The effort required depends on the type and license of the tool to be qualified. To this end in the following we have identified a number of different scenarios that are of relevance in the context of the openETCS project as depicted in Figure 3.

In the following considerations for the individual scenarios are described.

2.3.3.1 Self-developed Tool

For a self-developed tool the project needs to provide the means for qualification and quality assurance. As the tool has not been employed in productive use by others the “proven in use” argument (CENELEC EN50128 §6.7.4.4 a)) does not apply.

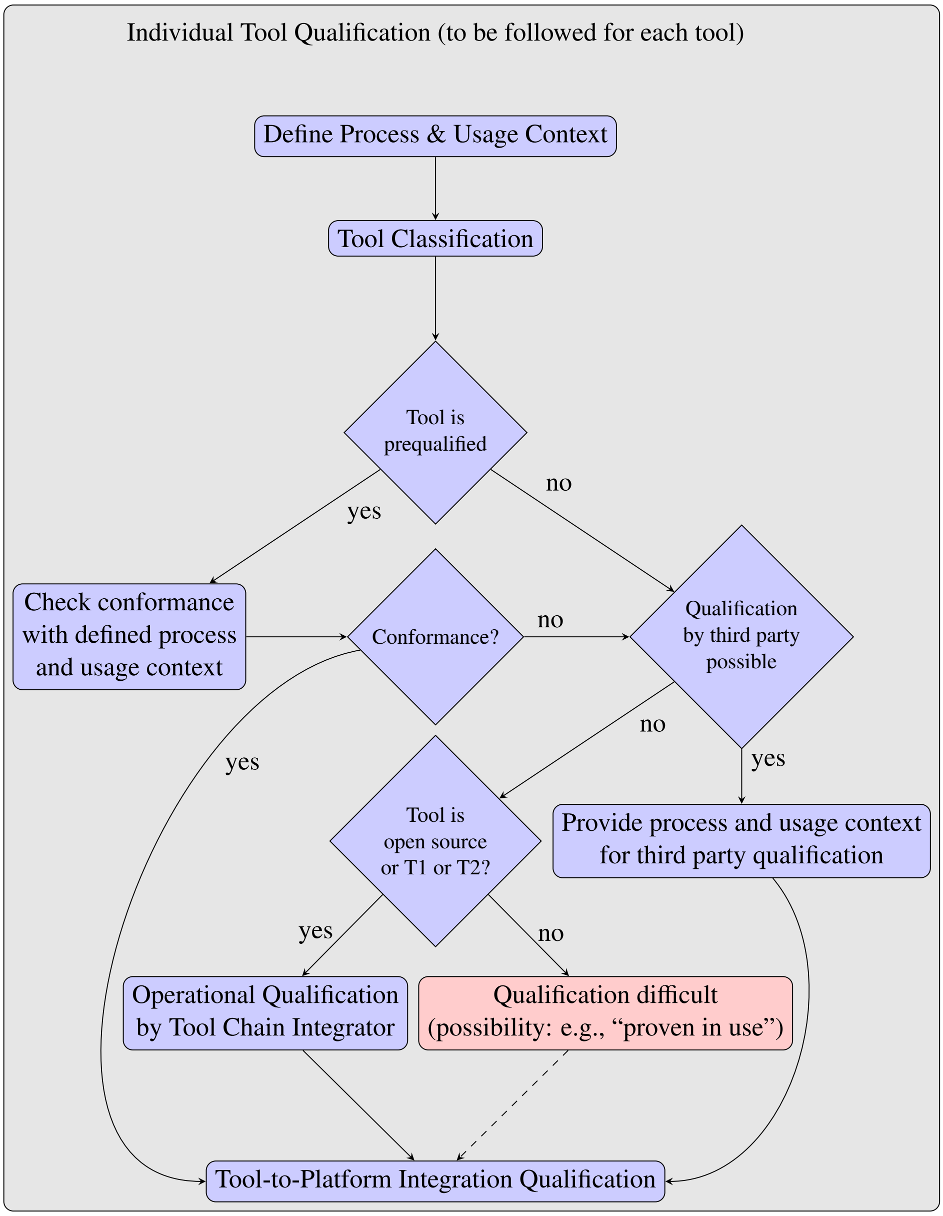


Figure 2. Proposal for the openETCS Tools qualification process

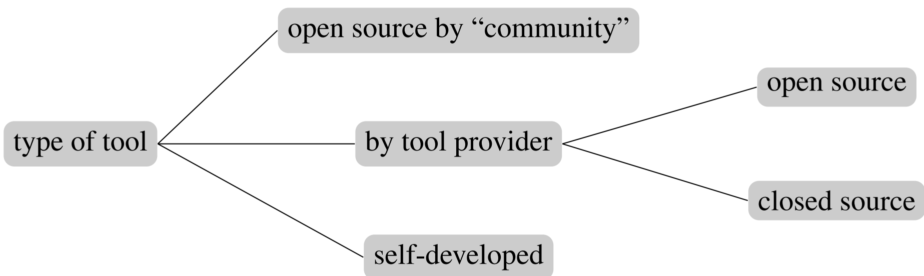


Figure 3. Scenarios to consider when qualifying individual tools

2.3.3.2 Tool Provided by a Tool Provider

For tools provided by a third party we must distinguish again two cases:

The tool is provided as closed source tool

If the tool is not pre-qualified by the tool provider the options are limited for T3 tools. If the tool provider is not willing or unable to provide the means for qualification, the qualification of a T3 tool will be difficult. One possibility is the “proven in use” argument: if the tool is already in use on a broad scale in industry and is generally regarded as reliable for the intended purpose this can be sufficient. This situation is represented as the red node in the process graph in Figure 2.

The tool is provided as open source

In this case it would be possible to adapt the tool and analyse it to enable qualification. However, this involves a huge amount of effort which is best done by the tool provider who as a better insight into the tool’s internal workings. Also a third party who has already experience with the tool or is a specialist with regards to qualifying safety-critical tools could be entrusted with the qualification task.

2.3.3.3 Open Source Tool Provided by the “Community”

If no tool provider can be identified but the tool is developed by the community and distributed under an open source license, the tool qualification has to be conducted by the project. However, many open source tools have a large development and expert community which can certainly be of help. In addition, the “proven in use” argument could possibly be applied (e.g., for tools like GCC).

2.3.4 Model-based Tool Qualification

This section describes a proposal on how to setup the operational qualification of individual tools. This step of the qualification process (cf. Figure 2) is required for any tool that is not pre-qualified. With respect to the scenarios described in Section 2.3.3 it applies to self-developed and third party tools classified T1 and T2 and open source T3 tools. In the case of closed source T3 tools it is not possible to follow this process as the internal workings of the tools are hidden.

We propose setting up the operational tool qualification based on the approach by Slotosch et al. which has been briefly sketched in Section 1.2.1. It has been simplified and adapted to our purposes. In particular, we will focus on the artifacts and documentation necessary to conduct the qualification steps. Figure 4 shows the different parts of the qualification model according to [10]. We propose to tailor this approach as follows:

- The different levels of requirements have been simplified: We propose to keep a single level of requirements. In addition we distinguish two special types of requirements: the *use cases* and *safety requirements* which are derived from a safety standard such as CENELEC EN 50128. Safety requirements often enforce constraints to the development process and the employed methods and do not necessarily specify desirable properties of the end result.
- In the context of openETCS the tool design model is built using SysML with the Papyrus tool.

- The “Quality Assurance” part, containing existing problem reports, their severity and relations to tests and potential errors, can be covered by using the GitHub issue tracker and is therefore not depicted in Figure 4.

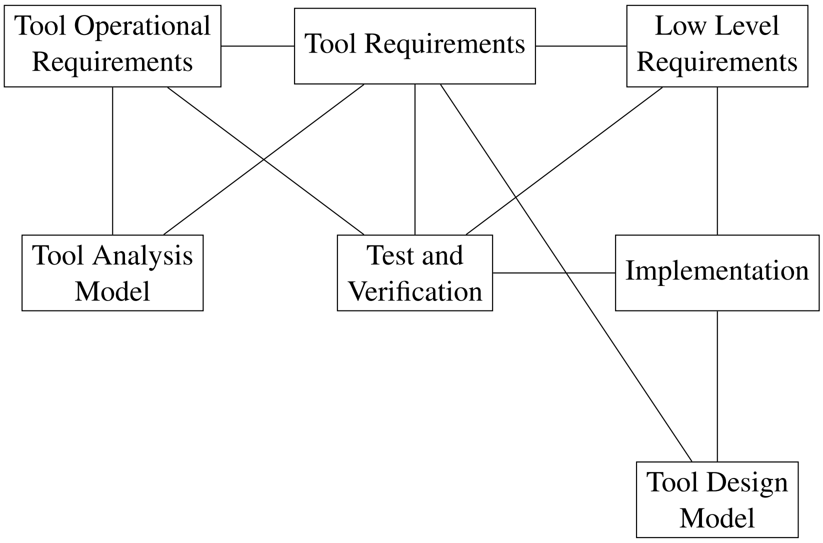


Figure 4. Parts of the qualification model and their linkage according to [10]

The SysML block diagrams depicted in Figure 5 depicts the proposed qualification meta-model for openETCS. It needs to be instantiated for each individual tool. The tool analysis model is used to analyse the functions of the tool and potential errors and mitigations. Ideally, it shall establish the traceability between requirements, use cases, implementations and test cases.

Each tool will then be defined by all information represented in our meta-model. This can be seen as our template for collecting all information needed for tool qualification.

2.4 Incremental Tasks for toolchain qualification

The qualification of the toolchain shall follow an incremental process adding the next tool to the previously qualified tool chain. The process will continue until the last qualified tool is integrated and the complete toolchain is qualified. A bottom-up approach will be applied. As a result, the errors related to tool interface should be easier to pinpoint.

A new OpenETCS tool chain release deal with different type of changes:

1. Add a new tool: the tool chain is extended
2. Replace a tool: a tool function is replaced by another one
3. Update a tool: a tool is update (bug fixes, performance improvements ...)

In the first case, the complete qualification process should be performed and the qualification of the new tool should be added.

In the two other cases, the feature model should also be updated, as well as the manual, the use cases and the potential errors. Moreover, evidence that all tools that depend on this tool are still compatible with the new one.

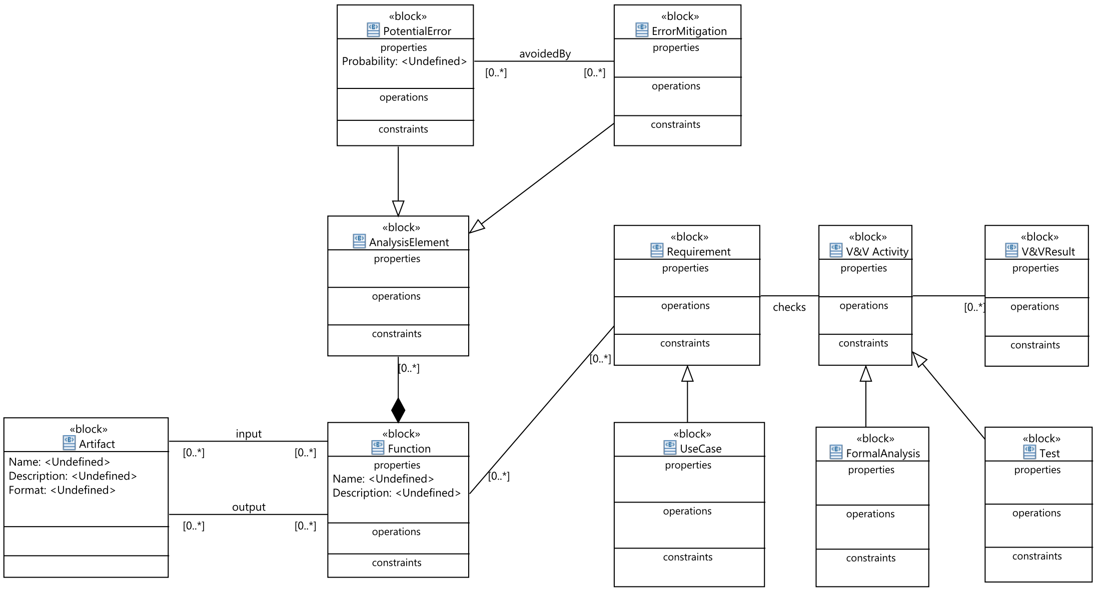


Figure 5. Tool analysis meta model for qualification

In all cases of a new qualification the three last phases should also be performed again.

During the toolchain qualification there will be iterative tasks where users repeat a set of actions over and over again. Thus, it would be necessary to evaluate these type of tasks and try to automate a subset of them to minimise the execution time.

2.5 OpenETCS Toolchain Qualification Process

This section will apply the set of concepts described in the previous section to the openETCS tool chain. Figure 6 implements the qualification process described in the previous section.

The first step is the **feature Analysis**. The OpenETCS tool chain will be defined by the set of its features and a guideline describing how to correctly use it. A SysML block diagram describes the tool chain architecture at a certain point in time as shown in Figure 7. This block diagram is intended to grow according to new feature requests and the needs of openETCS participants. This diagram will be kept updated as a reference of our tool chain. In any case, the complete information regarding the feature availabilities may be found in the Eclipse product definition.

Each feature of the toolchain is a block with the profile “openETCSFeature” and each artifact is block with the profile “openETCSArtifact”. Each feature realises at least one use case and may be implemented by one or more tools. Note that in the tool platform features may also be implemented as plug-ins. The diagram also imposes a (partial) order on the tasks. While some may be done in parallel, many tasks are dependent on others. Currently, the diagram neither highlights the use of the tools, nor the order of actions to be performed. This diagram should be completed by guidelines on how to use the tool chain and/or an activity diagram.

The second step, the **tool chain analysis** consists of Bundle Analysis and the work-flow Analysis. The openETCS tool chain is integrated into the Eclipse platform. One can analyses the sources of the tool chain and automatically derived the list of tools present in the tool chain as well as some information about them such as : the name, the version, a description and a list of dependency. These information should be included in the release documentation of the opentECTS tool chain.

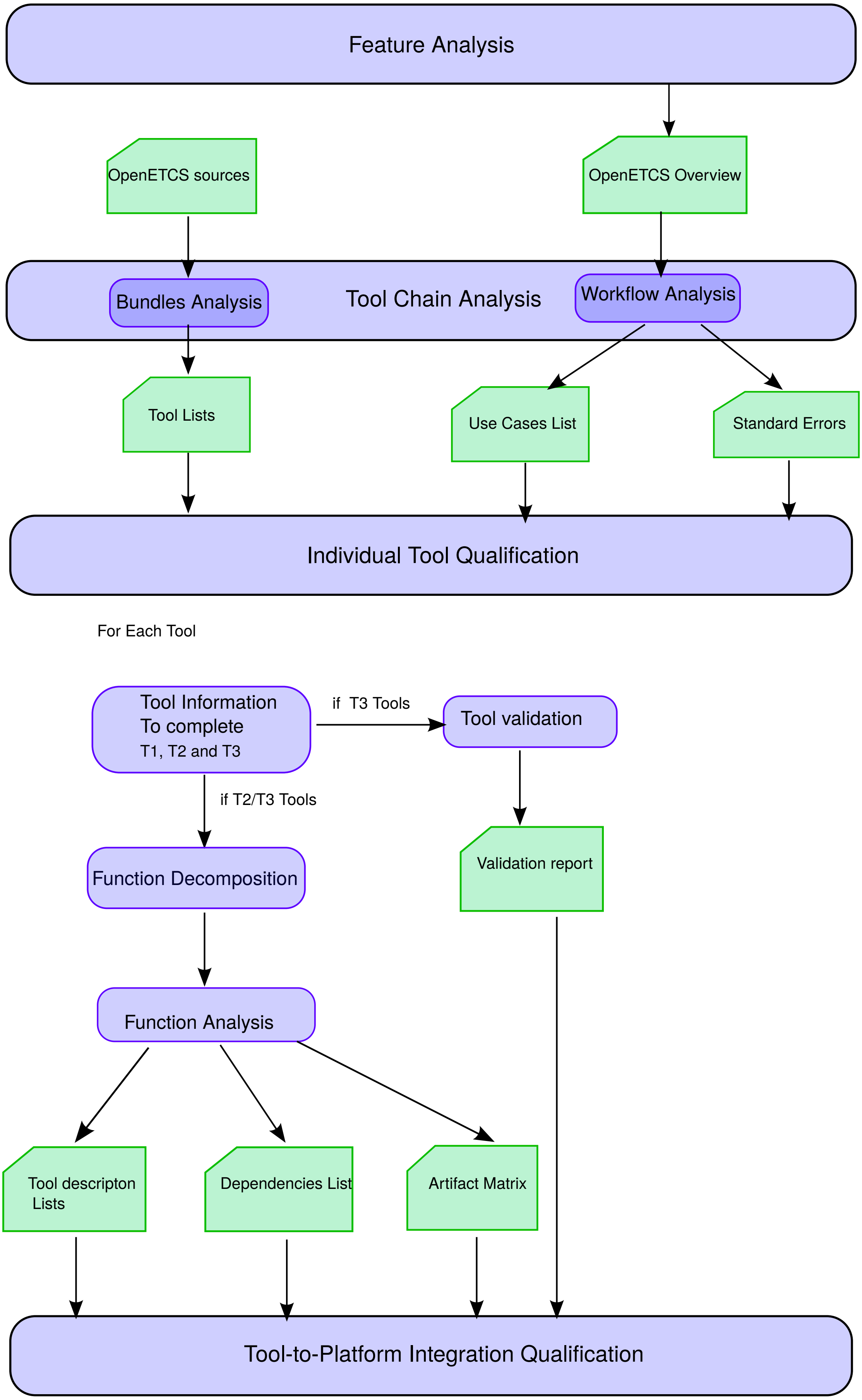


Figure 6. The OpenETCS Qualification process

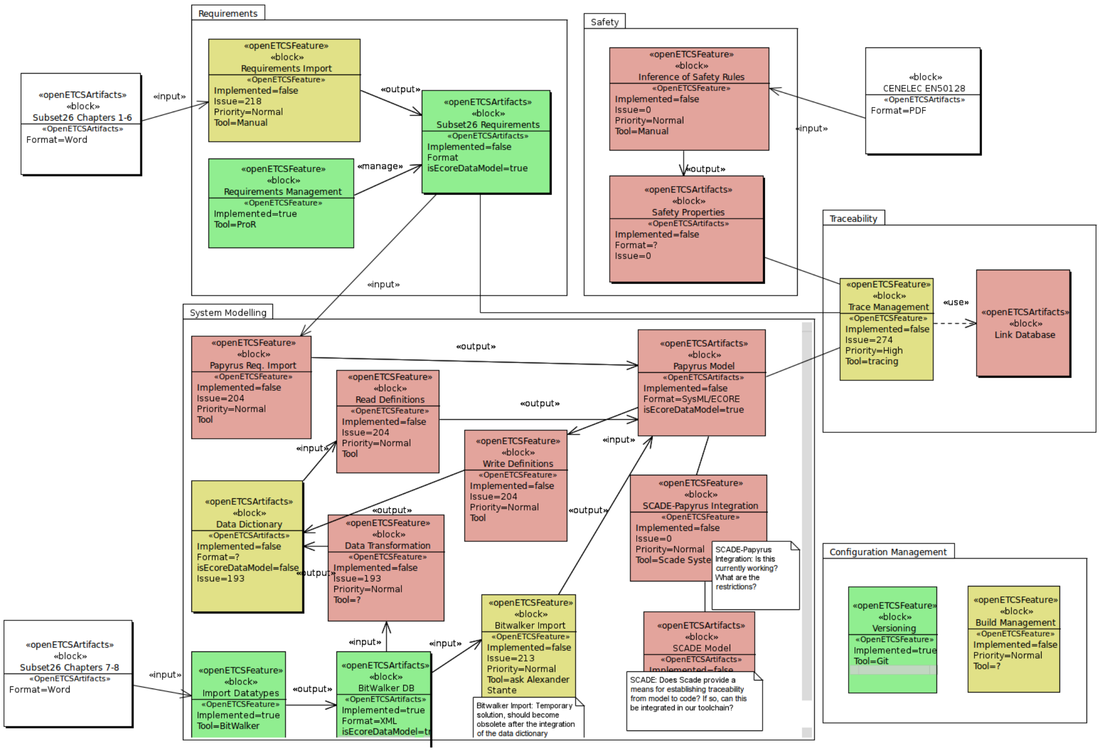


**Figure 7. Tool Chain overview (20.02.14) –
Green Block: Implemented
Yellow Block: Work in Progress
Red Block: Not started
White Block: External Artifacts**

The list of tool should be refined by the development team. The missing information should be added directly in the sources to make them available for the next tool releases.

From the work-flow of the tool chain it is possible to automatically derived some basic use cases such as parsing or producing artifacts. These basic use cases may be coupled with basic potential errors such as “no file found”, “too many file produced”...

After the tool chain analysis phase, the **individual tool qualification** starts. With the help of the tool list, the previous information given (from former releases) and the basic use cases, the tool information should be completed. Table 2 3 and 4 summarise the set of information needed for the qualification of a tool.

When all tool are completely defined the **tool-to-platform integration** starts. From the previous phases, we should be able to generate the dependency list of the tools and the artifact matrix. The dependency list is obtained by following the work-flow and the eclipse dependency list definition that includes the version of the tools. The dependency list is used to check if all tool can be correctly integrated into the tool platform by the qualification team and report to the development team. Part of this task may be automatically done by the Eclipse tool platform.

The artifacts matrix is the list of all artifacts produce within the tool chain with the link of each tool that uses, writes or reads it. As required by It allows the qualification team to check if all artifacts are correctly read and write and/or all possible errors are mitigated. In case of re-qualification of the tool chain, impact analysis of the change shall be perform form the artifacts matrix. Hence, the work may be alleviate by focusing on the tools that are impacted by the changes.

Table 2. Tool Information to be completed

Tool Name:	
Version:	
Tool License:	
Tool Origin:	
Tool Dependencies (with version)	 - -
Description:	
Tool Class:	
If T2/T3 Tools	
Tool Justification	
Manual or Specification Link and version	
Artifact List link	
Function List link	
If T3 Tools	
Evidence of correctness or failure detection [1]	
VnV activities report link	

[1] See EN50128 6.7.4.4 for the list of alternatives

Table 3. List of Artifacts Description

Name	Format	Is read ? By who ?	Restrictions	Time stamp check?	Is written ? By Who ?	Restrictions	Time Stamp produced ?
Artifact1	tex	Tool1	none	yes	no		
		Tool2	none	no	no		

Table 4. Function Definition

Name	
Description	
Inputs	
Outputs	
Use Case	

Potential Errors	Error Mitigation

3 Tool chain Qualification Example

3.1 Example - Consideration for two openETCS Tool Chain

In this section we will illustrate the qualification process for the openETCS tool chain. As an example we will focus on the sample tool cahin depicted figure 8

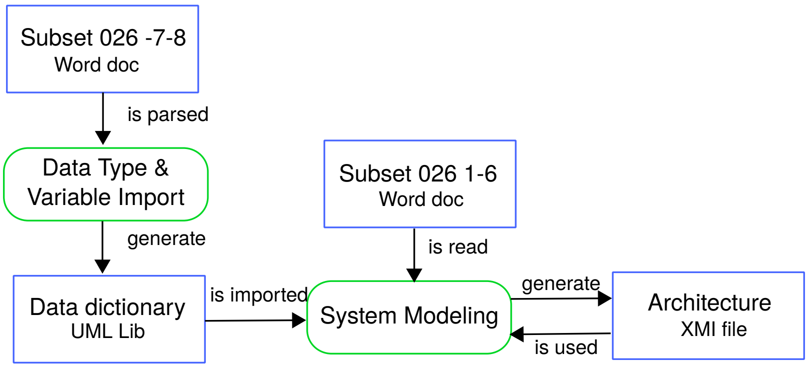


Figure 8. Tool chain sample

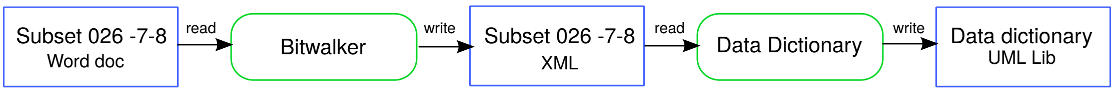


Figure 9. Feature: Data type and Variable Import

The first tool chain : Create High-level Architecture I proposes to create a high level view of the on board unit from the Subset 026 specification. It contains two features : Data type and Variable Import and the sytem modeling. Figure 9 describes the feature “Data type and Variable Import” in more details. The workflow is depicted figure 8.

The tool chain analysis give us the following list of tool :

1. Feature 1: Data type and Variable Import
   - Tool 1: Bitwalker
   - Tool 2: Data Dictionary import
2. Feature 2: System Modeling
   - Tool 1: Papyrus Editor
   - Tool 2: Papyrus SysML Checker

The following tables give the elements we could directly obtain from the eclipse analysis that have to be completed.

The artifact matrix for the tool chain matrix should look like table 9. This table has beed pre-filled with the information available now, this should also be completed during the qualification process.

Table 5. Data dictionary Information to be completed

Tool Name:	Data Dictionary
Version:	0.1.0
Tool License:	EUPL V1.1
Tool Origin Tool Dependencies	Fraunhofer ESK, LAAS-CNRS - org.eclipse.ui - org.eclipse.core.runtime - org.eclipse.emf.ecore - org.eclipse.papyrus.uml.extensionpoints
Description:	This feature contains the DataDictionary. The DataDictionary registers a UML model which contains data structure, variables and messages based on the ETCS System Requirements Specification (SRS).
Tool Class:	T3
If T2/T3 Tools	
Tool Justification	
Manual Link and version	
Artifact List link	
Function List link	
If T3 Tools	
Validation method	
VnV activities report link	

Table 6. Bitwalker Information to be completed

Tool Name:	Bitwalker
Version:	v1.0
Tool License:	EUPL v1.1
Tool Origin:	Siemens for openETCS
Tool Dependencies	None
Description:	Transform the word document chapter 7 and 8 to a XML file.
Tool Class:	T3
If T2/T3 Tools	
Tool Justification	
Manual Link and version	
Artifact List link	
Function List link	see 3.2
If T3 Tools	
Validation method	
VnV activities report link	

Table 7. Papyrus Editor Information to be completed

Tool Name:	Papyrus Editor
Version:	0.10.1.v20130918
Tool License:	Eclipse
Tool Origin:	Eclipse Incubation (CEA)
Tool Dependencies	- -
Description:	Papyrus is a modeling tool that allows us to model the high-level architecture of the OBU with SysML
Tool Class:	T1
If T2/T3 Tools	
Tool Justification	
Manual Link and version	
Artifact List link	
Function List link	
If T3 Tools	
Validation method	
VnV activities report link	

Table 8. Papyrus Checker Information to be completed

Tool Name:	Papyrus Checker
Version:	0.10.1.v20130918
Tool License:	Eclipse
Tool Origin:	Eclipse Incubation (CEA)
Tool Dependencies	- -
Description:	Papyrus checker verifies the basics rules of SysML modeling.
Tool Class:	T2
If T2/T3 Tools	
Tool Justification	To make import and export of the model the sysML model should be consistent with SysML rules.
Manual Link and version	
Artifact List link	
Function List link	
If T3 Tools	
Validation method	
VnV activities report link	

Table 9. List of Artifacts Description
The cells with ??? indicates unknown information

Name	Format	Is read ? By who ?	Restrictions	Time stamp check?	Is written ? By Who ?	Restrictions	Time Stamp produced ?
Subset 026 chapt.7-8	Word-docx	Bitwalker	none	no	no		
Subset 026 chapt.7-8	XML	Data dictionary	???	???	Bitwalker	???	???
UML Library	XMI	Papyrus	Papyrus XMI	???	Data dictionary	???	???
Papyrus Model	XMI	Papyrus	Papyrus XMI	???	Papyrus	Papyrus XMI	???

3.2 Bitwalker Model-Based Qualification Example

This section gives an example of a tool analysis model as described in Section 2.3.4 based on the specification import function of the Bitwalker tool.

Functions

FunctionBitwalkerSpecImport	
Name	Bitwalker Specification Import
Description	Parses the Subset 026 Word document and generates a the "Data Dictionary" representing data types, variables and messages
Inputs	ArtifactSubset026-7, ArtifactSubset026-8
Use Cases	UC1
Outputs	ArtifactDataDictionary
AnalysisElements	PotentialError1, PotentialError2, PotentialError3, PotentialError4, ErrorMitigation1, ErrorMitigation2, ErrorMitigation3

Artifacts

ArtifactSubset026-7	
Name	Subset 026 Word Document
Description	Subset 026-7 document containing the variable and data type definitions of the ETCS specification
Format	MS Word

ArtifactSubset026-8	
Name	Subset 026 Word Document
Description	Subset 026-8 document containing the message type definitions of the ETCS specification
Format	MS Word

ArtifactDataDictionary	
Name	Data Dictionary
Description	Data Dictionary representing data types, variables and messages
Format	Papyrus SysML

Use Cases

UC1	
Name	Subset 026 Transformation
Description	Generation of a Papyrus data dictionary from the Subset 026-7 and 026-8 documents
Actors	Papyrus Modeller
Steps	1. Selection of ArtifactSubset026-8 input file 2. Selection of ArtifactSubset026-9 input file 3. Indicate target Data Dictionary 4. Initiate transformation
Success Condition	The Data Dictionary contains exactly all definitions from the input documents; there are no deviations

AnalysisElements

PotentialError1	
Probability	High
Description	Variable / message / type not detected or missing in output
Mitigation	ErrorMitigation1

ErrorMitigation1	
Description	This error can possibly be detected and avoided as the modelling process is manual. Required in- or output described in the SRS can be added manually if missing.
Error	PotentialError1

PotentialError2	
Probability	Medium
Description	Variable or message have wrong type
Comment	This is a very dangerous error, especially in the case of different precision integers or floats. The error may remain unnoticed and be propagated to the code leading to potentially fatal malfunctions.
Mitigation	ErrorMitigation2

ErrorMitigation2	
Description	A mitigation is only possible by manual recheck (which would make an automatic conversion obsolete) of extensive/exhaustive testing or verification of the tool implementing the feature. However the inconsistent nature of the input document (Word) could prevent this.
Error	PotentialError2

PotentialError3	
Probability	Medium
Description	Missing fields in record / message
Mitigation	ErrorMitigation3

ErrorMitigation3	
Description	Can be detected if the functionality of the field is described in the functional description (similar to ErrorMitigation1).
Error	PotentialError3

PotentialError4	
Probability	Medium
Description	Wrong naming of variable / message / type
Mitigation	ErrorMitigation3

V&V Activities

Correctness Inspection	
Requirement	UC1/Success Condition
Description	Manually inspect whether all definitions of ArtifactSubset026-7 and ArtifactSubset026-8 are actually represented correctly in the Data Dictionary

Remark: It will be necessary to provide evidence that critical errors such as PotentialError2 or PotentialError4 can be detected or are not present in the tool. Exhaustive verification will be difficult due to the unreliable structure of the input document. The result must be correct also for ill-formed documents.

References

[1] Fredrik Asplund, Matthias Biehl, and Frédéric Loiret. Towards the automated qualification of tool chain design. In *Computer Safety, Reliability, and Security*, volume 7613 of *Lecture Notes in Computer Science*, page 392–399. Springer, 2012.

[2] Fredrik Asplund, Jad El-khoury, and Martin Törngren. Qualifying software tools, a systems approach. *Computer Safety, Reliability, and \ldots*, page 340–351, 2012.

[3] Sylvain Baro and Jan Welte. D2.6 requirements for openETCS. Requirements D2.6, openETCS, 2013. Cited by 0000.

[4] Matthias Biehl. Early automated verification of tool chain design. *Computational Science and Its Applications–ICCSA \ldots*, page 40–50, 2012.

[5] Matthias Biehl and M Törngren. Constructing tool chains based on SPEM process models. In *The Seventh International Conference on Software Engineering Advances (ICSEA2012)*, page 267–273, 2012.

[6] Biehl, Matthias, El-Khoury, Jad, Loiret, Frédéric, and Törngren, Martin. A domain specific language for generating tool integration solutions. In *In 4th Workshop on Model-Driven Tool & Process Integration*, 2011.

[7] Jörg Brauer, Jan Peleska, and Uwe Schulze. Efficient and trustworthy tool qualification for model-based testing tools. In Brian Nielsen and Carsten Weise, editors, *Testing Software and Systems*, volume 7641 of *Lecture Notes in Computer Science*, pages 8–23. Springer Berlin Heidelberg, 2012.

[8] Wen-Ling Huang, Jan Peleska, and Uwe Schulze. Test automation support. Delivrable D34.1, COMPASS, January 2013.

[9] Merlin Pokam and Norbert Schäfer. Report on CENELEC standards. Requirements D2.2, openETCS, April 2013.

[10] Oscar Slotosch. Model-based tool qualification : The roadmap of eclipse towards tool qualification. *Springer*, 2012.

[11] Oscar Slotosch, Martin Wildmoser, Jan Philipps, Reinhard Jeschull, and Rafael Zalman. ISO 26262-tool chain analysis reduces tool qualification costs. *Automotive 2012*, 2012.

[12] European Standard. *Railway applications-Communication,signalling and processing system- Software for railway control and protecton system*. CENELEC EN 50128. DIN, October 2011.

[13] Martin Wildmoser, Jan Philipps, and Oscar Slotosch. Determining potential errors in tool chains: strategies to reach tool confidence according to ISO 26262. In *Proceedings of the 31st international conference on Computer Safety, Reliability, and Security*, SAFECOMP'12, page 317–327, Berlin, Heidelberg, 2012. Springer-Verlag.