OETCS/WP7/D7.3

openETCS

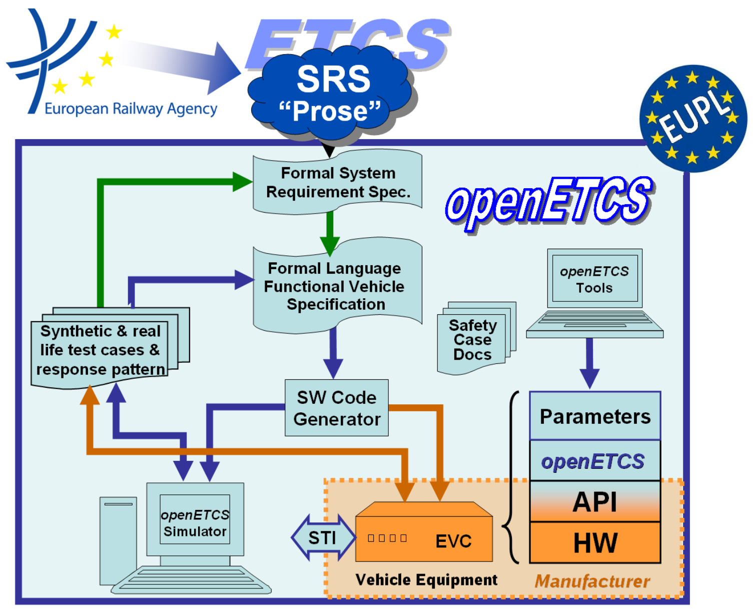
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 7: “Toolchain”

Toolchain Qualification Process Description

Cecile Braunstein and Jan Peleska

January 2014



Funded by:

Federal Ministry of Education and Research
Liberté • Égalité • Fraternité République Française Ministère de l'Enseignement Supérieur et de la Recherche
Investissements d'Avenir
Région de Bruxelles-Capitale
Gobierno de España
Ministerio de Industria, Energía y Turismo

Work-Package 7: “Toolchain”

OETCS/WP7/D7.3
January 2014

Toolchain Qualification Process Description

Cecile Braunstein and Jan Peleska
University Bremen

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	01.00
Document authors (org.)	Cécile Braunstein and Jan Peleska (Uni.Bremen)

Review information	
Last version reviewed	
Main reviewers	

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

1 Introduction to Toolchain Qualification

1.1 Tool Qualification

The CENELEC EN 50128 standard [11] 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 executable code;
- Tool class T2: Verification tools, the tool may fail to detect errors or defects;
- Tool class T3: Generated output directly or indirectly as part of the executable code.

The deliverable D2.2 [8] 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 [6, 7] 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.

In summary, the effort of qualification depends on the tool class 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. The tools Papyrus, SCADE and Bitwalker 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 RTTester 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.

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)

Tool	Purpose	Tool Class
Papyrus	Definition of the model architecture	T3
SCADE	Low-level modelling and 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

[9] 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 [10]: 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 [12] 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 increase 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 [5]. 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 [4, 3] 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 [3], 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 Toolchain Analysis

All the methods mentioned above start with a complete definition of the tool chain. In OpenETCS, the development of the toolchain follow an *agile* method, 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 incomplete toolchain and can re-use qualification information.

Moreover, as stated by Asplund and al., the toolchain provides some mechanism that has to be also ensured, reducing thereby the effort for each tool. These safety goals are related to the tool integration. In our context, most of the tool integration is made by integrated tools into a tool platform. From the previous cited paper, the tool platform should ensure the following safety-goals

- 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.

2.2 OpenETCS Toolchain Qualification Process

The OpenETCS tool chain will be defined by the set of its feature and a guidelines describing how to correctly use it. A SysML block diagram describes the tool chain architecture as shown figure 1.

This block diagram is intended to grow according to the new feature request and the needs of openETCS participants. This diagram should be kept updated as a reference of our tool chain, in any case the complete information regarding the features 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 realizes at least one use case and is implemented by at least one tool. Note that in the tool platform feature may also be implemented as plug-ins. The diagram also represents the order also shows which tasks maybe done in parallel and which ones are dependent of other tasks, but it does not highlight neither the use of the tools, nor the order of actions to be performed. This diagram should be completed by a guidelines on how to use the tool chain and/or an activity diagram.

To mitigate the qualification process, we will first consider more that one feature at a time considering that errors from a tool A may be detected by a Tool B in the next step of the tool chain process. Furthermore, the toolchain is a collection of feature and not tools, this differs from Asplund and al. in the sense that some of the tool integration mechanism that

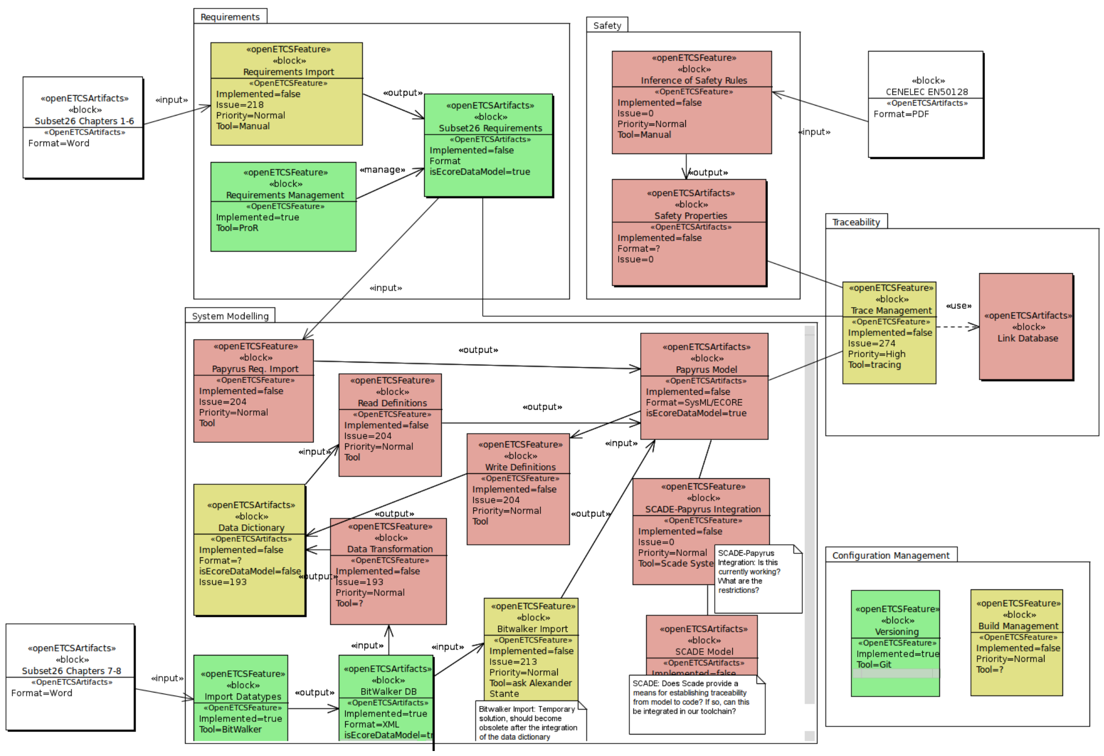

Requirements
«openETCSFeature» «block» Requirements Import «OpenETCSFeature» Implemented=false Issue=218 Priority=Normal Tool=Manual
«openETCSArtifacts» «block» Subset26 Chapters 1-6 «OpenETCSArtifacts» Format=Word
«input»
«output»
«openETCSArtifacts» «block» Subset26 Requirements «OpenETCSArtifacts» Implemented=false Format isEcoreDataModel=true
«openETCSFeature» «block» Requirements Management «OpenETCSFeature» Implemented=true Tool=ProR
«manage»
Safety
«openETCSFeature» «block» Inference of Safety Rules «OpenETCSFeature» Implemented=false Issue=0 Priority=Normal Tool=Manual
«block» CENELEC EN50128 «OpenETCSArtifacts» Format=PDF
«input»
«output»
«openETCSArtifacts» «block» Safety Properties «OpenETCSArtifacts» Implemented=false Format=? Issue=0
Traceability
«openETCSFeature» «block» Trace Management «OpenETCSFeature» Implemented=false Issue=274 Priority=High Tool=tracing
«use»
«openETCSArtifacts» «block» Link Database
«input»
System Modelling
«openETCSFeature» «block» Papyrus Req. Import «OpenETCSFeature» Implemented=false Issue=204 Priority=Normal Tool
«output»
«openETCSArtifacts» «block» Papyrus Model «OpenETCSArtifacts» Implemented=false Format=SysML/ECORE isEcoreDataModel=true
«openETCSFeature» «block» Read Definitions «OpenETCSFeature» Implemented=false Issue=204 Priority=Normal Tool
«output»
«input»
«openETCSFeature» «block» Write Definitions «OpenETCSFeature» Implemented=false Issue=204 Priority=Normal Tool
«input»
«openETCSArtifacts» «block» Data Dictionary «OpenETCSArtifacts» Implemented=false Format=? isEcoreDataModel=false Issue=193
«output»
«openETCSFeature» «block» Data Transformation «OpenETCSFeature» Implemented=false Issue=193 Priority=Normal Tool=?
«output»
«output»
«openETCSFeature» «block» SCADE-Papyrus Integration «OpenETCSFeature» Implemented=false Issue=0 Priority=Normal Tool=Scade System
SCADE-Papyrus Integration: Is this currently working? What are the restrictions?
«openETCSFeature» «block» Bitwalker Import «OpenETCSFeature» Implemented=false Issue=213 Priority=Normal Tool=ask Alexander Stante
«openETCSArtifacts» «block» SCADE Model «OpenETCSArtifacts» Implemented=false
SCADE: Does Scade provide a means for establishing traceability from model to code? If so, can this be integrated in our toolchain?
«openETCSArtifacts» «block» Subset26 Chapters 7-8 «OpenETCSArtifacts» Format=Word
«input»
«openETCSFeature» «block» Import Datatypes «OpenETCSFeature» Implemented=true Tool=BitWalker
«output»
«openETCSArtifacts» «block» BitWalker DB «OpenETCSArtifacts» Implemented=true Format=XML isEcoreDataModel=true
«input»
«input»
Bitwalker Import: Temporary solution, should become obsolete after the integration of the data dictionary
Configuration Management
«openETCSFeature» «block» Versioning «OpenETCSFeature» Implemented=true Tool=Git
«openETCSFeature» «block» Build Management «OpenETCSFeature» Implemented=false Priority=Normal Tool=?

Figure 1. Tool Chain overview (20.02.14) –
Green Block: Implemented
Yellow Block: Work in Progress
Red Block: Not started
White Block: External Artifacts

automates transformation of data, are part of the feature and are not falling out of the scope of the qualification.

Due to our development process, a "pre-qualification" of tools should be made when integrating a tool.

This high-level list should be detailed.

Tool Integration Process for Qualification

- Define name and version
- Describe use cases
- Provides justification of the tool within the tool chain
- Provide input/output artifacts format (associated with the version)
- Integrate the tool in the SysML model
- Provide tool manual and other available documentation (associated with the version)
- Link with an issue tracker

One possible implementation is to represent all these in formations directly in the SysML model.

This high-level list should be detailed. What I would expect: Which artefacts exist; How they are connected; When artefacts have to be

The Qualification Process

1. Feature Analysis
   - This step should assign a class to each feature based on the use cases.
   - Define the potential errors
   - Identify counter measure and/or error detection
   - For T3 tools 2 alternatives: certified compiler/generator or object code checker and/or exhaustive tester
2. Tool platform analysis
   - Provide evidence of the safety-goals mentioned in the previous sub-section
3. Toolchain Analysis
   - Defines the work-flow
   - Identify the “hot spots” of the toolchain
   - Rearrange the toolchain if possible
   - Find new measures when needed with combining tools (redundancy with orthogonal codes . . .)
4. Toolchain qualification verification
   - Check consistency of tool version with manuals and other provided feature information
   - Generate table to check if all possible errors has a detection or a correction mechanism
   - Generate the qualification report

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

[4] 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.

[5] 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.

[6] 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.

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

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

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

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

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

[12] 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.