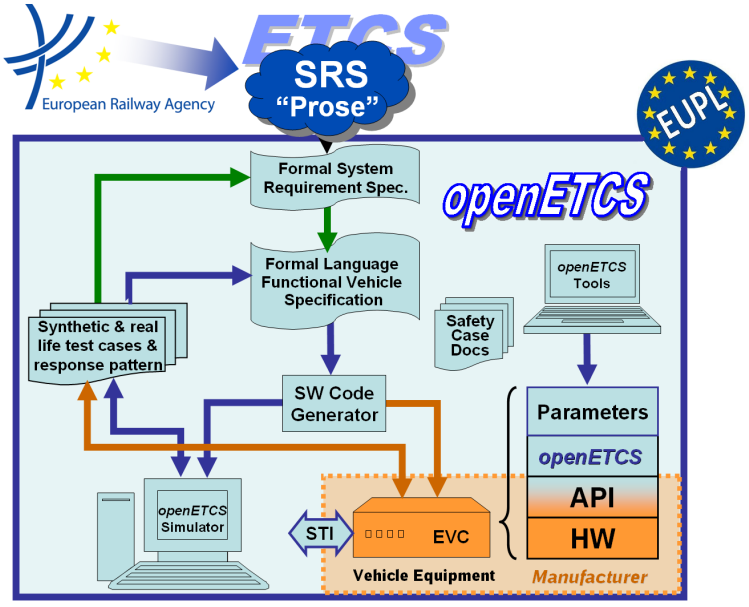
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 7: “Primary tool chain”

Report on the final choices for the primary toolchain

Decision on the final choice for the means of description (O7.1.4), tools (O7.1.8) and tool platform (O7.1.11)

Marielle Petit-Doche and all participants of the decision process July 2013



Funded by:

Federal Ministry of Education and Research
Région de Bruxelles-Capitale
GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

This page is intentionally left blank

Work-Package 7: "Primary tool chain" **OETCS/WP7/D7.1 – 00/03**
July 2013

Report on the final choices for the primary toolchain

Decision on the final choice for the means of description (O7.1.4), tools (O7.1.8) and tool platform (O7.1.11)

Marielle Petit-Doche

Systerel

all participants of the decision process

WP7 partners

Deliverable

Prepared for openETCS@ITEA2 Project

Abstract: This document gives a description and the results of the first task of WP7. The objectives of the task are to analyse and recommend, means, tools and platform to develop the primary tool chain of Open ETCS.

Table of Contents

Figures and Tables

Figures

Tables

Document information	
Work Package	WP7
Deliverable ID or doc. ref.	D.7.1 (including O7.1.4, O7.1.8, O7.1.11)
Document title	Report on the final choices for the primary toolchain
Document version	00.03
Document authors (org.)	Marielle Petit-Doche (Systerel) and WP7 partners

Review information	
Last version reviewed	00.0x
Main reviewers	

Approbation			
	Name	Role	Date
Written by	Marielle Petit-Doche Michael Jastram Cécile Braunstein Uwe Steinke Stanisla Pinte Alexander Stante	WP7-T7.1 Sub-Task Leader	
Approved by	Michael Jastram	WP7 leader	

Document evolution			
Version	Date	Author(s)	Justification
00.01	08/07/2013	M. Petit-Doche	Document creation
00.02	18/07/2013	M. Jastram, C. Braunstein	Complements in § 3
		U. Steinke	Draft version of appendix A
00.03	19/07/2013	M.Petit-Doche	Complements in § 2 and § 4

1 Introduction

The aim of this document is to report the results of the task T7.1 of WP7 : "Primary tool Chain analyses and recommendations".

1.1 T7.1 Objective

The goal of WP7 is to provide other openETCS workpackages with tooling for their activities. Tools have been separated into primary and secondary tooling. The objective of this deliverable is to define the best primary tooling, considering all constraints of the project. Selecting tools also entails selecting modeling languages and an integration platform.

1.2 Scope of T7.1

The scope of the primary tooling has been defined in the WP7 Description of Work []. Figure 1 depicts the main process (from D2.3). In the scope of the primary tooling are (according to the DoW):

Sub-System Semi-formal model. "A semi-formal model of the system specification is defined from the SSRS. This model shall reflect the architecture defined in SSRS. (...) The semi-formal model shall be as consistent as possible with the SSRS level of abstraction, in particular choices concerning software architecture and design have not to be described at this level. In practice, all the requirements of SSRS and of the sub-system Hazard analysis shall be covered by the semi-formal model." (D7.3, 4.4.3). "The means of description of the semi-formal

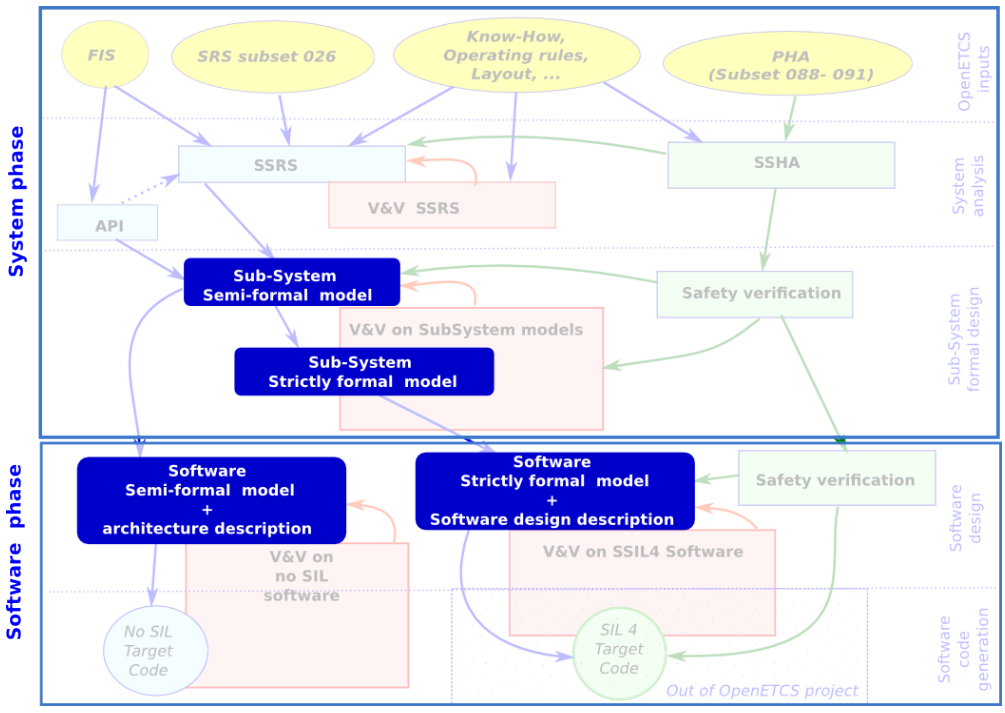


Figure 1. Main OpenETCS process. The models that are covered by primary tooling are shown in blue.

model shall be understandable by domain experts, providing graphical description" (D7.3, 4.4.4).

Sub-System Strictly formal model. "This semi-formal model *can* be extended with strictly formal models to improve the understanding of some part of the sub-system." (D7.3, 4.4.2). "To facilitate safety activities, the safety relevant function should be as much as possible insulated from non safety relevant functions." (D7.3, 4.4.3).

Software Semi-formal model + architecture description. The main output of this step is a semi-formal model which allows to produce executable code. This model shall be completed by a Software Architecture and Design Specification, which describes the software architecture and the design choices. (...) The semi-formal model defined during the system phase shall be completed, keeping the same language or extending it to cover specific software aspects. (D7.3, 4.5.2).

Software Strictly formal model + Software design description. This model is concerned with the functional and safety branch. This activity "shall provide methods and a toolchain to obtain SIL4 executable code of the on-board software application." (D7.3, 4.5.3).

1.2.1 Scope with respect to SSRS, API and code

API, Code and SSRS[1] are in the scope of the secondary toolchain (T7.2). Of course, choices in the primary tool chain that may affect the secondary tool chain will be covered in this document.

The SSRS poses a special challenge, as activities in its creation have already started. Even though no tools have been selected yet. Deliverable D2.3 describes the form of the SSRS as follows: "The SSRS (...) shall be described as textual documents." (4.3.4). It continues to state: "However these documents shall be completed by a semi-formal model to describe the functional architecture of the on-board unit".

The connection between SSRS and Sub-System Semi-formal model is described as: "A semi-formal model of the system specification is defined from the SSRS. (...) In practice, all the requirements of SSRS and of the subsystem Hazard analysis shall be covered by the semi-formal model."

There is even less information regarding the API, except that it is handled corresponding to the SSRS: "The SSRS and API shall be described as textual documents. However these documents shall be completed by a semiformal model to describe the functional architecture of the on-board unit." (D2.3, 4.3.4).

There is little relevant information with regard to code generation in this report, except that it makes some implications regarding the software functional model: "A first executable code is produced from the software functional model. This executable code shall be non vital. However it shall be able to run in real time on a on-board computer. Thus it shall comply to the standardised interfaces." (D2.3, 4.6.2).

[1] Note that the DoW does not even mention the SSRS, as its creation has been proposed the first time in February 2013, when the DoW was already finalized. Therefore, we include SSRS-related activities with T7.2.3 (Requirement traceability) and report on them in O7.2.5 (Requirement management tool choices).

1.3 T7.1 activities

The activities have started in November 2012, with a proposal of benchmark organisation. After selection of a set of case studies (specified in D2.5 [1]), different approaches have been proposed and models have been stored on a common open github repository. All the methods have been presented during a public meeting in April 2013.

Besides, a set of criteria have been defined according the D2.6-9 requirement document [2]. The results are recorded in the outputs O7.1.3-O7.1.7 [3] for means and tools and O7.1.9 [4] for tool platform.

A decision meeting took place the 4th of July 2013 to analyse the results of the benchmark and to decide which means and tools will be retained during the process.

Results of the decision are given in this current document.

1.4 Glossary

API Application Programming Interface

DoW Description of Work. In this document we typically mean the WP7 DoW.

FIS Functional Interface Specification

HW Hardware

I/O Input/Output

OBU On-Board Unit

PHA Preliminary Hazard Analysis

SIL Safety Integrity Level

SRS System Requirement Specification

SSHA Sub-System Hazard Analysis

SSRS Sub-System Requirement Specification

SW Software

V&V Verification & Validation

2 Results on means and tools for primary tool chain

2.1 Proposed Tool Chains

There were three tool chain proposals in total (Figure 2. These are:

Scade. A Scade-based primary toolchain would consist of the two tools Papyrus and Scade. An integration between the tools already exists, and both tools cover all activities. The biggest advantage is that there would be little additional work necessary to cover the primary tool chain. The biggest drawback of this solution is the fact that Scade is not open source.

ERTMS Formal Specs (EFS). An EFS-based primary toolchain would mainly consists of the Papyrus, EFS and additional components from the Eclipse ecosystem, looking at Polarsys for guidance. It is not clear if and how the formal models could be modelled with this tool chain. The biggest advantage of this approach is that it is open source, and that a significant portion of Subset 26 has already modelled with EFS. The biggest drawback is that it is not clear how the integration with Papyrus would look like, that EFS is only partly ported to Eclipse, and that it is not clear how laborious the tailoring of the Eclipse-based formal modeling tools would be.

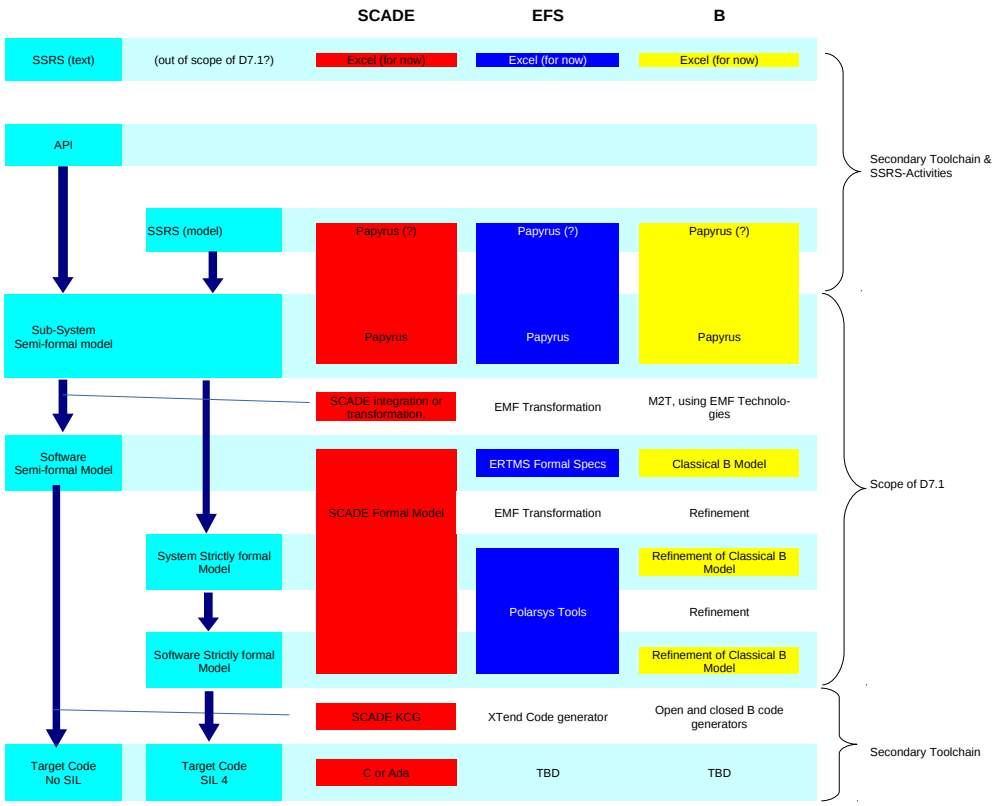


Figure 2. Proposed Tool Chains

B. In contrast to the previous proposals, this one starts bottom-up, starting with the assumption that code generation from B models is possible and practical. This tool chain proposes working with B on the bottom two layers, but leaves open how these are connected to the Papyrus-based top. The biggest advantage of this approach is that the resulting model will be usable, and that there is a rich existing ecosystem for B, both open source and commercial. The biggest drawback is that there are many blanks to fill in, which may require significant development work.

2.2 Initial list of candidates

The initial list of candidates is the following:

1. GOPRR
2. CORE
3. ERTMSFormalSpecs
4. SysML with Papyrus
5. SysML with Entreprise Architect
6. SCADE
7. EventB with Rodin
8. Classical B with Atelier B
9. Petri Nets
10. System C
11. UPPAAL
12. Why3
13. GNATprove

For each approach and tool, the initial author of the evaluation is the partner in charge of the modelling. Two assessors, for each approaches, are in charge of the review of the evaluation and can correct it or add comments. For each approaches, the models are available on the public github `https://github.com/openETCS/model-evaluation/tree/master/model` (see 3). Scores of each approaches according the evaluation criteria are record in appendix of the outputs O7.1.3-O7.1.7 [3].

2.3 Evaluation results

In the conclusion part of O7.1.3-O7.1.7 [3], the first table (see 4) show how the evaluated approaches cover the openETCS design process:

The highest score is 9 and means that the criteria is fully respected, the lowest score is 0. The higher scores (more than 6) for each approach is graphically represented on figure 5.

The second table (see 6) in the conclusion part of O7.1.3-O7.1.7 [3], shows that all evaluated approaches, except GnatProve, are adapted to modeling and design activities:

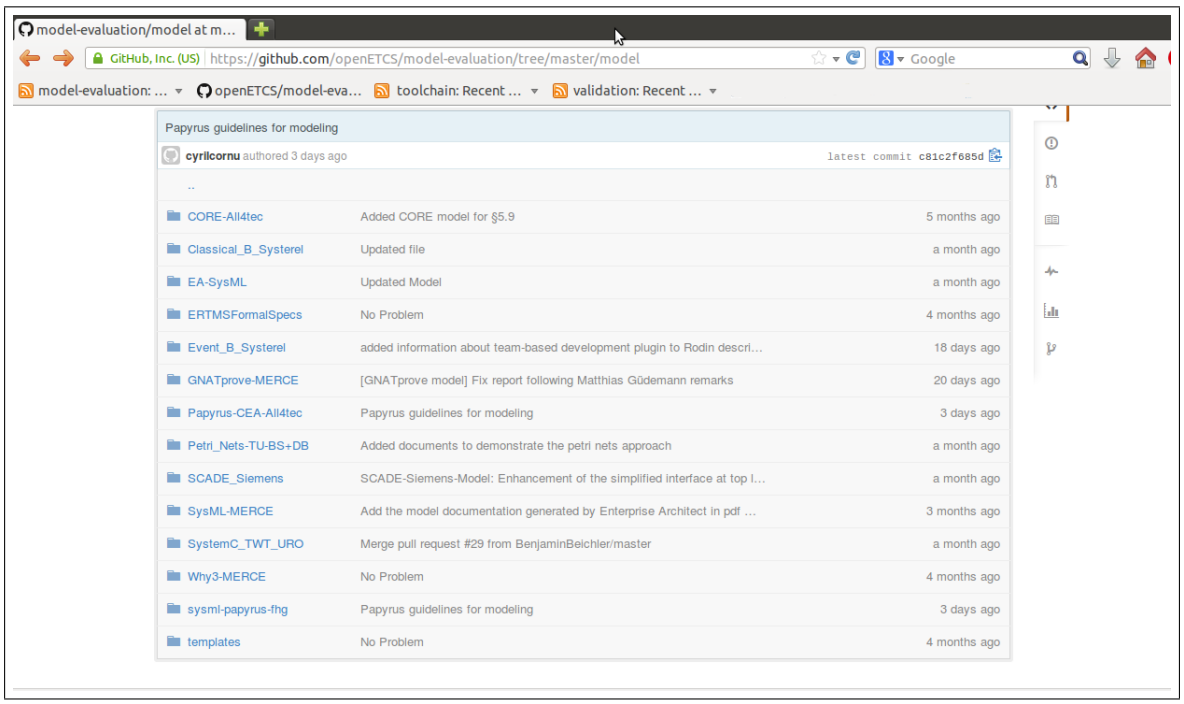


Figure 3. Repository of models

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with EA	SCADE	EventB	Classical B	System C	Petri Nets	GNATprove
System Analysis	5	1	7	**9**	3	**9**	3	2	**6(9)**	2 (3)
Sub-system formal design	**9**	**9**	6	7	**9**	**9**	5	5	**6(9)**	3 (4)
Software design	**9**	0	6	7	**9**	6	**9**	**9**	**6(9)**	**6(9)**
Software code generation	**9**	0	3	3	**9**	3	**9**	6	2 (3)	**6(9)**

Figure 4. Use of the approaches during process phases

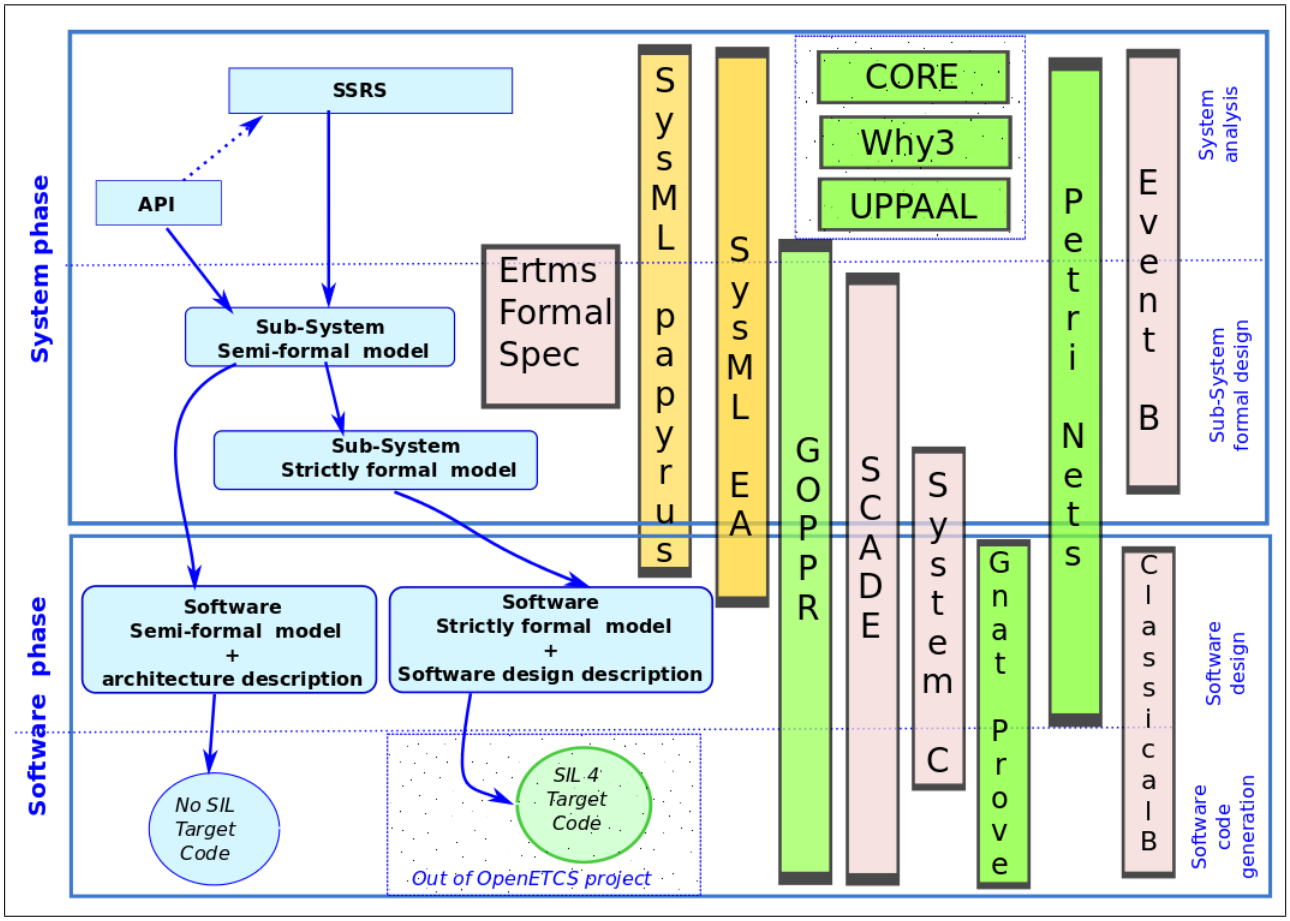


Figure 5. Results of candidates

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with EA	SCADE	EventB	Classical B	System C	Petri Nets	GNATprove
Documentation	3	7	6	7	8	7	0	0	2 (3)	2 (3)
Modeling	**9**	**9**	**9**	**9**	**9**	**9**	**9**	8	**6(9)**	2 (3)
Design	6	**9**	6	7	**9**	7	8	**9**	5(7)	3 (4)
Code generation	**9**	1	3	4	**9**	3	**9**	5 *	2 (3)	**6(9)**
Verification	0	7	6	3	8	**9**	**9**	4 *	**6(9)**	**6(9)**
Validation	0	**9**	5	4	8	**9**	4	7	**6(9)**	**6(9)**
Safety analyses	0	0	4 *	6	1	6	3	3 *	5(7)	2(3)

Figure 6. Use of the approaches according activities

2.4 Short list

The evaluation process has not been completed for three approaches:

CORE The tool is not open source and difficult to obtain, it seems possible to cover the same task with an open-source approach as SysML

Why3 Gnat-Prove covers at least the same service and seems more efficient

UPPAAL It is a tool dedicated to the verification and validation of time-constraints properties, for example joined with SystemC. It has been proposed for the benchmark on secondary tools (T7.2).

At the end of the evaluation, three others approaches have been discard from the primary tool chain :

GOPPR SysML seems a better candidate to offer the same services.

GnatProve According the results (see 6), GantProve has been proposed to joined the evaluation of secondary tools (task T7.2).

PetriNets However it is a well-known formal approaches, more recent approaches seems more adapted to the goals of the project.

Two tools associated to the SysML approach have been proposed to the evaluation: Papyrus and Enterprise Architect. After discussion on pros and cons of each, partners agree that Papyrus covers better the objectives of the project, especially the open-source requirement.

Due to the evaluation and discussion during the decision meeting, the partners agreed on a short-list of approaches (see 7):

1. ERTMSFormalSpecs
2. SysML with Papyrus
3. SCADE
4. EventB with Rodin
5. Classical B with Atelier B
6. System C

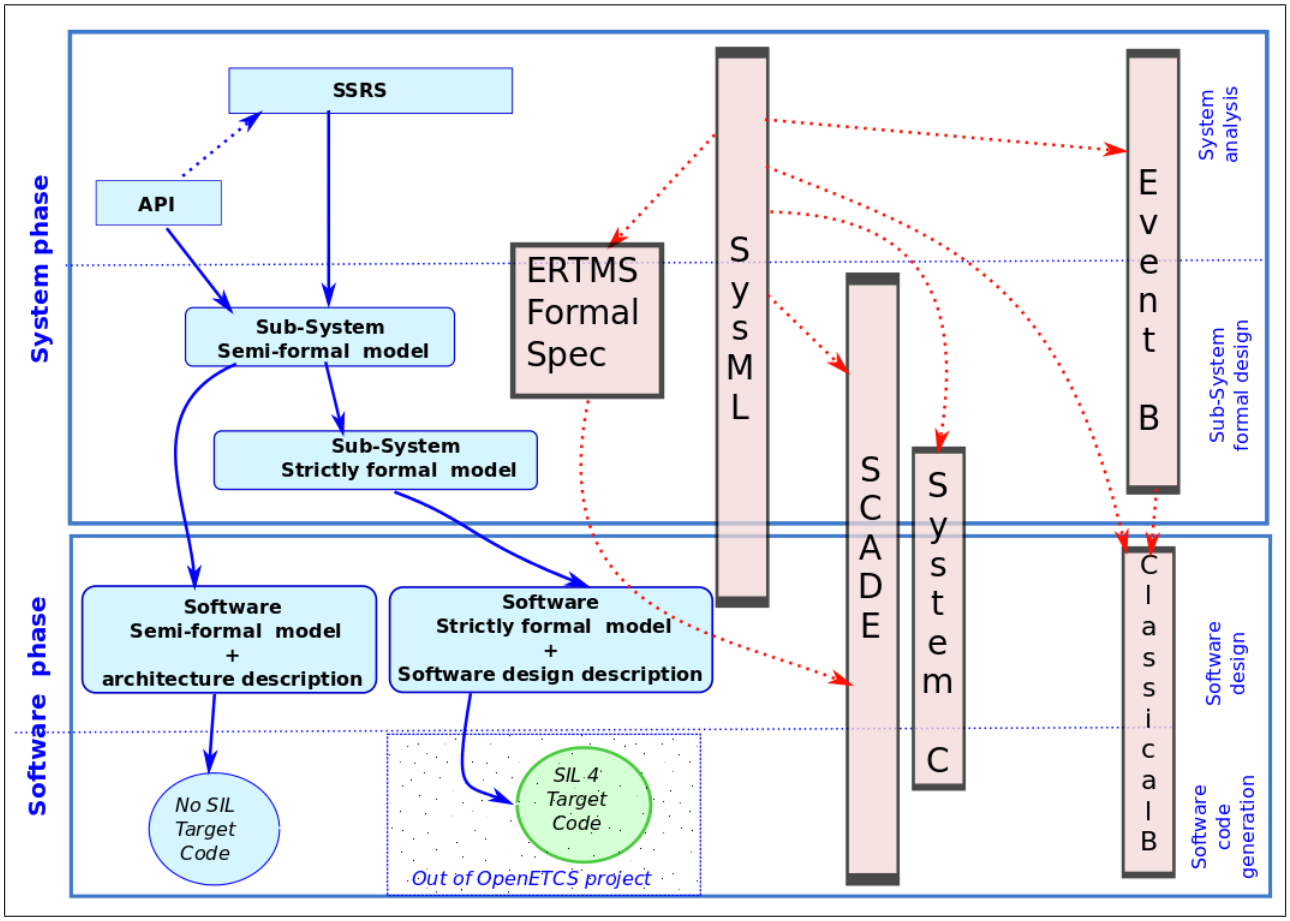


Figure 7. Short list of candidates

3 Results on tool platform

The tool platform should provide mechanisms to integrate various tools. The tool platform is not the primary nor secondary tools, nor the tool chain. It is the support for the tool chain implementation, it shall help to integrate the tools into a seamless tool chain. The evaluation will focus on the integration capabilities of the tool platform.

Todo: Description of the candidates by Cecile Braunstein

3.1 Initial list of candidates

- Eclipse
- TopCased/Polarsys
- RTP-Cesar
- Mono/.NET
- SCADE

After a first round, Mono/.NET and SCADE were discard because they do not comply to our tool platform definition. RTP-CESAR was also discard, the maturity of this project is not yet usable. Finally, **Eclipse with the modeling framework (EMF)** has been chosen as a tool platform, the possibility to use Polarsys and take some part of the TopCased tool chain as well as which version of Eclipse and EMF to choose are discussed in the next sections.

Eclipse can also integrate framework, It has also be decided that any framework added to the Eclipse platform within the OpenETCS tool chain should be documented (version, usage ...) and clearly justify.

3.2 Eclipse

Eclipse is an open source Tool Platform originally developed at IBM. It has been explicitly designed as an extensible platform to enable different tools to exchange data and share common functionality. Additionally Eclipse is a rich open source ecosystem with a variety of frameworks for different purposes, such as versioning, code generation, language support and many more. The Eclipse Modeling Framework (EMF) as a top level project bundles all modeling frameworks at Eclipse. Additionally it technically provides a common data format for modeling purposes. Originally it has implemented the OMG Standard Meta-Object Facility (MOF) and has then be reduce to the OMG standard essential MOF (eMOF). EMF provides model-driven approach to develop modeling languages. It allows to define custom meta-models and generate code form them. Additionally it provides common features such as command-based editing and XMI serialization for generated models. In the following we show how Eclipse and EMF aligns with the openETCS requirements.

Open Source

All Eclipse core components including EMF are open source and under the Eclipse Public licenses, which allows for commercial use and is compatible to the EUPL. The Eclipse Foundation and the Eclipse Development process assure the management of the intellectual properties for all Eclipse projects. Additionally all Eclipse projects follow a common infrastructure and process allowing external partners to contribute and maintain projects.

Long-Term Maintenance

The Eclipse Foundation also provides infrastructure and a process for Long-Term Maintenance for all Eclipse projects. It enables users of a technology to contract service providers to maintain current and older versions of these technologies. These service providers do not necessarily have to be committers on the original projects.

Portability

Eclipse itself is implemented in Java and therefore portable to all major operating systems. The underlying UI technology SWT is implemented for all major and even most uncommon window kits. As SWT uses native widgets, the performance of the UI is close to native applications. The Eclipse Java IDE has a user based of several million developers, which ensures, that the platform runs stable on the supported platforms. Since version 4.2, EMF is part of the core platform. However, EMF does not contain any OS specific components and is therefore highly portable.

Tools Interoperability

The Eclipse Platform has been explicitly designed to enable various tools of the software lifecycle to collaborate. It provides mechanisms, such as a service oriented architecture and extension points to enable the communication between different parts of a tool chain. EMF is well-suited as a common data-format. The collaboration of a large number of tools is shown and validated in the various Eclipse packages, which are released in the yearly release train.

Modularity

Eclipse is based on OSGi, a standard for modularization of Java applications. The Eclipse OSGi runtime Equinox is the reference implementation of OSGi. OSGi enables to modularize a system, in this case the tool chain. Additionally it allows to specify the API of modules and the dependencies between them. Additionally, the existing platform provides many possibilities to be extended by new features. The extensibility and OSGi as an underlying technology allow fully customizing the Eclipse Platform. Existing pieces and frameworks can be added to a tool chain, new parts can be developed.

Framework Support

Over the last ten years, a rich ecosystem of frameworks has developed around the Eclipse Platform. All these frameworks are developed under the EPL and checked for IP cleanliness. Eclipse frameworks cover all different kinds of purposes, however there is a strong focus in support for tool development and modeling. Modeling technologies are almost all compatible with EMF as a common data format. Technologies provided by Eclipse projects include:

1. Textual Modeling and DSL (e.g. Xtext)

2. Language Support (e.g. CDT, JDT)
3. Source Code Versioning Clients (e.g. Egit, Subclipse, Subversion)
4. Model Repositories and Versioning (e.g. EMFCompare, EMF Diff/merge, EMFStore and CDO)
5. Code Generation (e.g. Xpand, Xtend)
6. Model Transformation (e.g. ATL, QVT)
7. Model Development Tools (e.g. Papyrus, OCL, RMF, Sphinx, eTrice)
8. Graphical Modeling (e.g. Graphiti, GMF)
9. User Interfaces (e.g. JFace, Databinding, EMF Client Platform, EEF)
10. ALM Tooling (e.g. Mylyn)

3.3 Version Management

3.4 Topcased and Polarsys

Topcased is a tool for systems engineering, based on Eclipse and various Eclipse projects. Polarsys is a project concerned with the long term support of the Topcased tool chain. There is an overlap between Topcased and the openETCS tool chain. There is also an overlap between the objectives of openETCS and Polarsys:

Topcased and openETCS tool chain. Both, Topcased and the openETCS tool chain are based on Eclipse. Further, the openETCS tool chain will definitely use Papyrus, which is also part of Topcased. And last, both are concerned with covering all aspects of the V-Model, although for different domains (aviation vs. rail).

Polarsys and openETCS. The objectives of Polarsys and openETCS overlap significantly as well: Both are concerned with tools in a safety-critical domain, requiring tool qualification, etc. They are also concerned with long term support through open source.

3.4.1 State of Topcased and Polarsys

While the state of the art document mentions Topcased [], it was not evaluated as a whole. Merely the Papyurus component of Topcased was evaluated, but a newer version than the one used by Topcased.

Topcased is using a fork of an old version of Papyrus (Ver. 0.8.2) which is no more supported by the CEA (actual version 0.10.X) and, as the CEA is not part of Topcased, no more code development over this version/Topcased will be done by CEA. Unfortunately, the development on Topcased modeler (forked version of Papyrus) is not so active anymore: 60 commits on the 3 last months (as of July 2013) against more than 1600 commit for Papyrus. Further, the actual version of Papyrus have been greatly improved with respect to stability since version 0.8.2, and some stability issues may have not been corrected in Topcased.

To conclude, Topcased requires Eclipse 3.7.2 Indigo (1.5 year-old version) which is no more supported by the Eclipse foundation. Some part of Topcased initiative (plugins/add-ons) may still be very useful to the openETCS project, so we will reach out to the Polarsys community to

see whether there is an interest in aligning versions for long term support. The versions currently used in Topcased are not suitable for the openETCS tool chain, unfortunately.

4 Decision

This chapter gives the decision done during the meeting of the 4th of July 2013.

4.1 Decision on the tool platform

By vote, all the partners agree on the use of **Eclipse** as tool platform.

The last release is Kepler 4.3, launched the 26 of June 2013.

4.2 Decisions for high level step

By vote, all the partners agree on the use of the **SysML** approach supported by the **Papyrus** tool to cover the higher level of the OpenETCS V-cycle.

SysML 1.1 is completely supported on Papyrus, 1.2 is largely supported, moreover not implemented part of the norm may be added through motivated requests. The last release of Papyrus is 0.10.0, launched the 26 of June 2013 and is based on Eclipse Kepler 4.3.

4.3 Propositions of approach to cover all the design process

No vote has been done during the meeting on the approaches to cover the lower levels of OpenETCS V-cycle. Three propositions of chain are still under evaluation:

- a combination of **SysML and Scade** described in A
- a combination of **SysML, ERTMSFormalSpec and Topcased** described in B
- a combination of **SysML and Classical B** described in C

For the moment, all the candidates of the short-list (see figure 7) are still take in consideration for example to cover lack of the main tool chain or verification and validation purposes.

4.4 Conclusion

Todo: To complete if necessary.

References

[1] Guillaume Pottier David Mentre, Stanislas Pinte and WP2 participants. Methods and tools benchmarking methodology. Technical Report D2.5, OpenETCS, 2013.

[2] Sylvain Baro and Jan Welte. Requirements for openETCS. Technical Report D2.6, OpenETCS, 2013.

[3] Marielle Petit-Doche and WP7 partners. Evaluation of the means and tools against the WP2 requirements. Results O7.1.3-O7.1.7, OpenETCS, 2013.

[4] Cécile Braunstein and WP7 partners. Evaluation of tool platforms against the WP2 requirements. Results O7.1.9, OpenETCS, 2013.

[5] Michael Jastram, Marielle Petit-Doche, Jonas Helming, and Jan Peleska. openETCS toolchain WP7 descritpiton of work. Defin D01, OpenETCS, February 2013.

[6] Rico Kaseroni. Project quality assurance plan. Technical Report D1.3.1, OpenETCS, 2013.

[7] Marielle Petit-Doche and Matthias Güdemann. openETCS process. Technical Report D2.3, OpenETCS, 2013.

[8] Merlin Pokam and Norbert Schäfer. Report on CENELEC standards. Technical Report D2.2, OpenETCS, 2013.

[9] Jan Welte and Hansjörg Manz. Report on existing methodologies. Technical Report D2.1, OpenETCS, 2013.

[10] Hardi Hungar. Report on v&v plan and methodology. Technical Report D4.1, openETCS, 2013.

[11] Jani Welte. Safety plan. Technical Report O 4.4.1, openETCS, October 2013.

[12] Klaus-Rüdiger Hase. Project outline full project proposal annex openETCS. Technical Report v2.2, openETCS, 2011.

[13] UNISIG. SUBSET-026 – System Requirements Specification. Technical Report 3.3.0, ERA, March 2012.

[14] UNISIG. SUBSET-076 – Test related ERTMS documentation (this version is related to version 2.3.y of SUBSET-026). Technical Report 2.3.y, ERA.

[15] UNISIG. SUBSET-088 2.3.0 - ETCS Application Levels 1 & 2 - Safety Analysis. Technical Report 2.3.0, ERA.

[16] UNISIG. SUBSET-091 3.2.0 — Safety Requirements for the Technical Interoperability of ETCS in Levels 1 & 2. Technical Report 3.2.0, ERA.

[17] UNISIG. SUBSET-034 3.0.0 — Train interface FIS. Technical Report 3.0.0, ERA.

[18] Comission Decision. CCS TSI for HS and CR transeuropean rail. Technical Report 2012/88/EU, EU, January 2012.

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

[20] European Standard. *Railways applications — The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) — Part 1: Basic requirements and generic process*. CENELEC EN 50126_1. DIN, January 2000.

[21] European Standard. *Railway applications — Communication, signalling and processing systems — Safety related electronic systems for signalling*. CENELEC EN 50129. DIN, May 2003.

Appendix A: SysML and Scade

Todo: *Description of the approach by Uwe Steinke.*

Diagram A1 illustrates the most important components and operational relationships of a system and software modelling toolchain based on SysML, Papyrus, SCADE and Eclipse. All components and links shown with solid lines are available, while the dashed ones are intended to be implemented within the openETCS project.

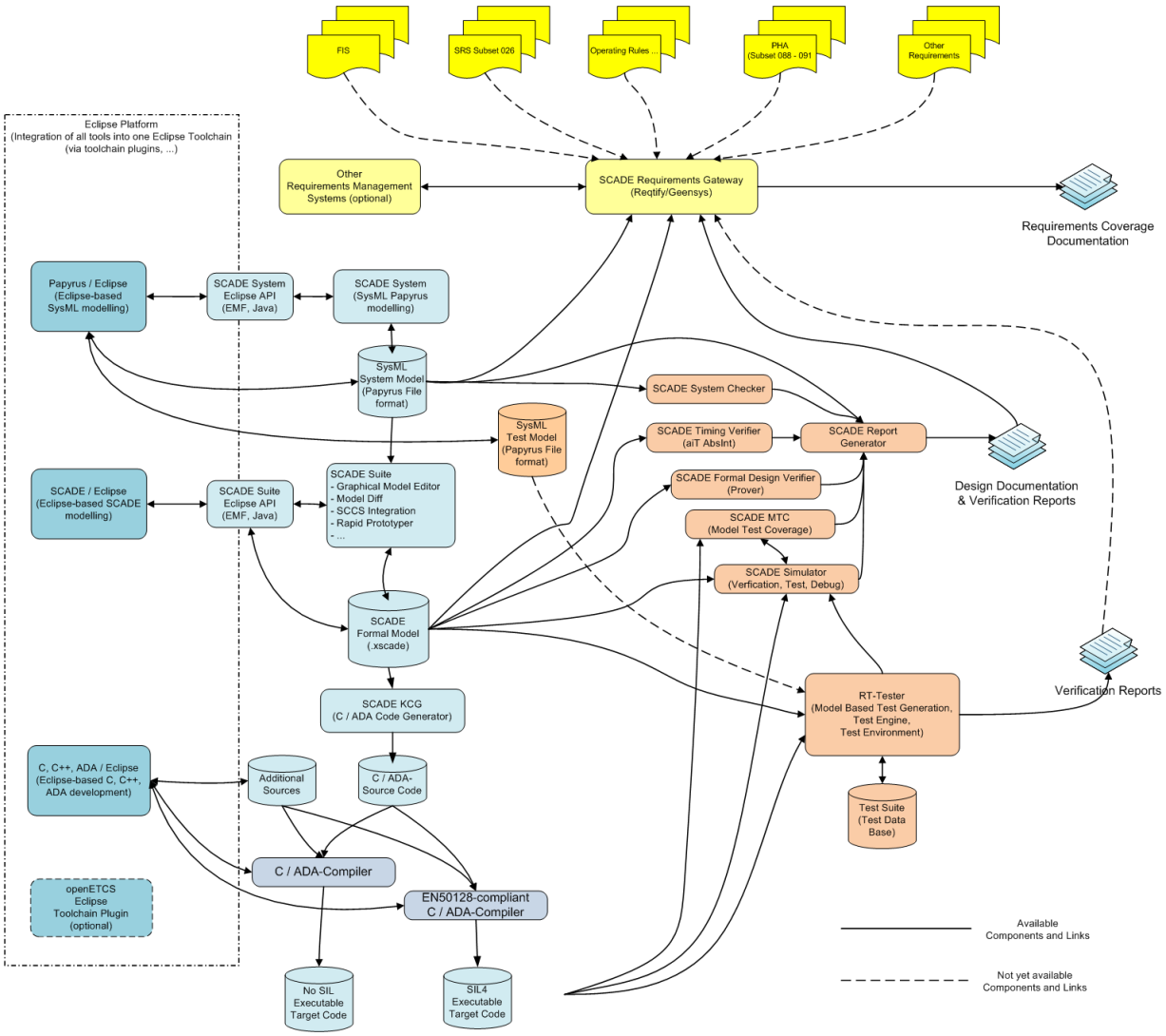


Figure A1. SysML SCADE Toolchain

The diagram colors are chosen related to the colors in 1:

- Requirements and requirements management components in yellow
- "Blue" openETCS design process (see 1) elements in light and dark blue

- Eclipse is painted dark blue
- Verification elements in red

Within the following paragraphs and subsections a short description of this approach will be given by walking through the tool chain and the design process.

A.0.1 Requirements management

The SCADE Requirements Management Gateway is based on Reqtify from Geensoft / Dassault Systems and serves to collect and link all requirements from the openETCS input documents and related objects as design und verification documents, model and source code artefacts, test cases, test protocols etc. It supports impact analyses and generates requirements traceability and requirements coverage reports. If needed for the openETCS process, it can be complemented with other requirement management systems and already comes with interfaces to these. ProR for example could be integrated in this way.

A.0.2 Semiformal System and Subsystem Modelling with SCADE System / Papyrus

SCADE System is an integration of Papyrus into the SCADE IDE intended for SysML system modeling. It allows to modelize the interactions and hierarchical dependencies between the various parts of a complex system through design elements representing functions, data and interfaces.

The idea is to model system structures, data types / data dictionaries, inputs, outputs, interfaces and relationships between blocks with SysML and transfer it to native SCADE for behavioral modelling automatically. Since Papyrus and SCADE System are using the same file formats, there is no prevention of using all SysML capabilities that Papyrus supports, but in this case without automatic transfer to native SCADE.

SCADE System supports SysML Block Definition Diagrams (BDD) and Internal Block Diagrams (IBD).

More details about the relationship between Papyrus and SCADE System:

1. SCADE System incorporates Papyrus, but SCADE System is more than Papyrus, as outlined in the following list.
2. SCADE System enhances Papyrus with functionalities on top:
   - Report Generator: Generation of design documents out of the SysML model
   - Model Check: user expandable checker on SysML models (OCL rule checking, . . .)
   - Model Diff: Semantic comparison of models
   - Interface to requirements management (Reqtify)
   - Synchronization with SCADE Suite
3. The idea of SCADE System is to ease the work for system and software architects. SCADE Systems users don't need so deep knowledge and experience of UML, SysML and . . . profiles as native Papyrus requires, so that architects or rail engineers, that are not UML experts, are able to work with. Therefore, SCADE System tailors Papyrus in several aspects:

- Cleaned up views and user interface and less chances of doing things wrong compared with native Papyrus.
- Support for designing blocks (BDD, IBD), relationships and data flows/interfaces between them (as Papyrus).
- Support for data types, data dictionaries (as Papyrus).
- Allocation of functions (as Papyrus).

4. Actually, SCADE System does not support behavior modeling, but has it on the road map for the next year.

5. SCADE System uses the same file formats as Papyrus, so that SCADE System and Papyrus files are exchangeable in both directions. SCADE System simply hides SysML artefacts it does not support. This gives the opportunity for:

   - Using SCADE System for non-UML/SysML experts (architects and rail engineers)
   - Using Papyrus for UML/SysML experts software experts

6. For integration with Eclipe, SCADE System (as well as SCADE Suite) comes with an Eclipse EMF API including the enhancements on top of Papyrus

With the exception of the synchronization with SCADE Suite, the mentioned easements and enhancements may also be achievable with Papyrus by tailoring and added functionality. For openETCS, behavioural modeling seems to be necessary for a rather complete model on SysML level. Therefore, SCADE System, because ready to use right now, could be chosen for architectural modeling and SSRS matters at the beginning and then continued with Papyrus, when a tailored Papyrus variant becomes available.

A.0.3 Formal Modelling with SCADE Suite

SCADE Suite integrates all modelling, verifation and supporting SCADE tools under the roof of the SCADE IDE. The components relevant for descending part of the development "'V"' process are

- SCADE Suite Editor (Graphical and textual modelling)
- SCADE Requirements Gateway Integration (Linking of model artefacts with requirements)
- SCADE Model Check (model syntax check)
- SCADE Model Diff (model comparison)
- SCADE Simulator (graphical debugging, simulation and testing)
- SCADE Rapid Prototyper (quick control and display elements for rapid prototyping, optional)
- SCADE Code Generator KCG (C / ADA code generation)
- SCADE Reporter ((Design) report generator)

The most important tools for modelling are editor and code generator. The others mentioned are mainly verification tools, but very useful and practically indespensible for agile development.

At least, to cover all elements of the "blue" design process in diagram 1 a C / C++ / ADA compiler is required. For building not safety-relevant executables any C-Compiler (gnu c, ...) is suitable, for safety relevant executables the compiler must be compliant with EN50128.

A.0.4 Model Verification

The verification of openETCS SCADE models can be performed with the "'red"' components shown in diagram A1. Most of them are part of SCADE System or SCADE Suite:

- SCADE Simulator: The SCADE Simulator should be used in an agile iterative development process for a steady accompanying verification of the modeling work. Simulator test scripts allow executing verificaton suites for the models automatically. Via its automation and co-simulation interface it is able to be integrated in the openETCS tool chain.
- SCADE Model Test Coverage MTC: The MTC serves to determine structural model test coverage while executing test scenarios. Instead of measuring code coverage on the generated code, it works on the model structure directly. The MTC tool is automatable.
- SCADE Design Verifier: The verifier performs formal proving and bases upon a model checker from from Prover Technology AB.
- SCADE Timing Verifier: Execution timing verificaton based on aiT from AbsInt.
- SCADE Report Generator: Generation of design reports for SCADE Suite and Scade System model als well as for verification results.
- RT-Tester: Test engine and environment with model based test case generation and interface to SCADE, see openETCS contributions of the University of Bremen. The ability to derive test cases from a test model complements the verification tool chain with model based testing capabilities.

A.1 Description of the approach for OpenETCS design process

Todo: How the proposed approach covers all "blue" design process (see 1) ?

The approach as specified in the previous subsections (Chapt. A0) (insert ref xxx) covers all elements of the "blue" design process (see 1) by using the SCADE tool chain including requirements management, semiformal system and formal subsystem/software modelling, code and executable generation. An Eclipse integration is provided (see following Chapt. xxx).

A.2 Integration of the approach with SysML/Papyrus

Todo: How the proposed approach can be integrated with the SysML/ Papyrus selected for the high level of design process ?

Because SCADE System bases on Papyrus mounted into the SCADE IDE and uses native Papyrus file formats, a seamless integration with SysML / Papyrus is available. Models can be edited with both, Papyrus and SCADE System and therefore exchanged in both directions.

Behavioral modeling has to be done with Papyrus until supported by SCADE System as announced for the near future.

A thrilling question for the openETCS process might be, if and - if yes - which of the artefacts on system level should be modelled with SysML, that can not be transferred to native SCADE automatically.

A.3 Integration of the approach with Eclipse

Todo: How the proposed approach can be integrated with the Eclipse, selected as platform for OpenETCS tool chain ?

All SCADE tools can be run and controlled via command line and/or via automation interfaces.

SCADE System (SysML modelling) and SCADE Suite (SCADE modelling) already come with Eclipse API plugins based on EMF. These enable to access (read and modify) the model project information, meta and model data from within Eclipse. The plugins additionally display the model structure, but they don't show the model graphics in Eclipse.

If graphical modelling should be done within Eclipse, this has to be implemented by openETCS. It is in doubt, if the effort for this activity would be applicable; without any effort, the SCADE editor should be used instead.

Nevertheless, the provided Eclipse integration is worthwile to supply all openETCS users, that are not directly working on the SCADE models, with an integrated Eclipse tool chain. The idea of such an integration is to have one build tool chain, that starts and runs an openETCS executable build process with one button click beginning from all (heterogeneous) sources and performing all necessary model transformations, code and executable generation. This could be achieved with an "'openETCS Eclipse Tool Chain Plugin"', implemented as part of the openETCS project with the goals ease-of-use and convenience.

In summary, an Eclipse integration is available. An optional "'openETCS Eclipse Tool Chain Plugin"' could improve the convenience for openETCS tool chain users.

A.4 Benefits versus OpenETCS requirements

Todo: Discuss the benefits in regards of OpenETCS requirements and expected results.

The most important benefits of the SysML/SCADE approach are:

- seamless integration,
- completeness,
- maturity,
- qualification for safety critical development,
- productivity
- availabilty just now.

The SysML/SCADE approach covers almost all aspects of the openETCS process and lets expect to fill gaps with manageable effort.

Therefore, the modelling work for openETCS can begin immediately.

Additionally, the SCADE language covers the capabilities of the ERTMSFormalSpec language, so that ERTMSFormalSpecs models could be transferred to SCADE automatically. Nevertheless, this would require a model transformator, that does not exist actually.

A.5 Shortcommings versus OpenETCS requirements

Todo: Discuss the shortcommings in regards of OpenETCS requirements and expected results.

The SysML/SCADE approach has one drawback: the tools are mainly not open source. These facts may help to better come to terms with it:

- The SCADE language is documented and very regular.
- The file formats are documented and easy to understand.

Therefore, the SysML/SCADE approach is open for bidirectional transformations to other modeling languages.

A.6 On going work for openETCS project

The availability of the nearly complete SysML/SCADE tool chain gives the freedom to focus on the few items to clarify:

Todo: Which are the elements to clarify, to specify or to develop, in order the approach suit the openETCS process ?

How can we evaluate and plan this work ?

which skills is needed ?

The availability of the nearly complete SysML/SCADE tool chain gives the freedom to focus on few items to clarify:

- Since the tool chain offers several capabilities and options, it has to be determined how these shall be used within the openETCS process.
- A justifiable balance has to be found between semi-formal modeling in SysML and formal modeling in SCADE.
- Some aspects of the RT-Tester integration into the tool chain has to be clarified in detail.
- A requirements management has not been set up for openETCS up to now. If ProR is chosen, it has to be interfaced with the shown Requirements Management Gateway with little effort.
- An "‘openETCS Eclipse Tool Chain Plugin"’ should be implemented (for convenience only).

A.7 Conclusion and other comments

The most challenging question is, how deep the openETCS functionality should be modeled semi-formal and when to start with formal modeling. The question could be answered best if focusing on adequacy: A justifiable balance of technical and non-technical aspects as feasibility, complexity, efficiency, overall effort, project schedule etc..

Using SysML/Papyrus with SCADE offers two different alternatives for semiformal modeling:

1. Use SysML/Papyrus for semi-formal modeling by utilizing only the SysML language artefacts, for which an automatic transformation to SCADE exists today. The remaining – formal – modeling then has to be done with SCADE. Advantage: The interfacing between SysML/Papyrus is done, the tool chain already complete and ready for modeling right now. Disadvantage: The semi-formal SysML model will not be as comprehensive as the second alternative 2; there will be system aspects that only reside within the SCADE model, but not in the SysML model.

2. Use SysML/Papyrus for modeling all system aspects as far as justifiable with respect to understandability, maintainability, adequacy and effort. It requires the determination of a suitable subset of SysML to avoid the model becoming unrulable. Advantage: This approach leads to a most complete model on SysML level. It allows to benefit from future transformations between SysML and appropriate formal modeling languages, as soon as they may become available. Disadvantage: The openETCS project has to implement the transformation tools from SysML to formal modeling languages; for SCADE, that applies to all SysML artefacts not supported by the existing transformators.

In summary, alternative 1 is easy and ready to use and needs little effort. Alternative 2 offers more flexibility on SysML level but needs more effort and time. At least, finding a suitable balance between technical and non-technical aspects could answer the question.

The fact, that the SysML/Papyrus approach already exists and is operable, offers the chance to start the openETCS modeling process just now without delay. In parallel, a truely complete open source and open proof openETCS tool chain can be set up without causing inacceptable impact on the modeling work until it becomes mature enough for practical usage.

Appendix B: SysML, ErtmsFormalSpec and Eclipse/Polarsys

Todo: Description of the approach by Stanislas Pinte.

The proposed approach combines three tools existing today to provide an integrated toolchain, from system design to code generation.

B.1 Description of the approach for OpenETCS design process

Todo: How the proposed approach covers all "blue" design process (see 1) ?

The blue boxes of the overall design process are covered in the following manner:

SSRS box: Using Papyrus for modelling the high-level system design in SysML language.

Sub-system semi-formal model box: Using ERTMSFormalSpecs to model the complete SSRS into a semi-formal model.

Software semi-formal model + architecture description box: Shall be done inside Eclipse/Polarsys tools for transforming a semi-formal model into software source code. The exact mix of Eclipse/Polarsys tools is to be decided by the participants of task T3.8.

Target source code box: This box is covered by the source code generated by the Eclipse/Polarsys tools. This source code can then be compiled and executed on the demonstrator hardware.

B.2 Integration of the approach with SysML/Papyrus

Todo: How the proposed approach can be integrated with the SysML/ Papyrus selected for the high level of design process ?

The proposed approach uses SysML/Papyrus as top-level component. Moreover, as SysML/Papyrus and ERTMSFormalSpecs both support an EMF-based interface, technical integration between both tools is not an issue.

As ERTMSFormalSpecs is already implementing 44% of Subset-026, some questions are open:

- How can the SysML system-level model be connected with the ERTMSFormalSpecs semi-formal model? - Can a SysML system-level model be generated based on the existing 44% of Subset-026, so that this SysML system-level model can then be improved,

B.3 Integration of the approach with Eclipse

Todo: How the proposed approach can be integrated with the Eclipse, selected as platform for OpenETCS tool chain ?

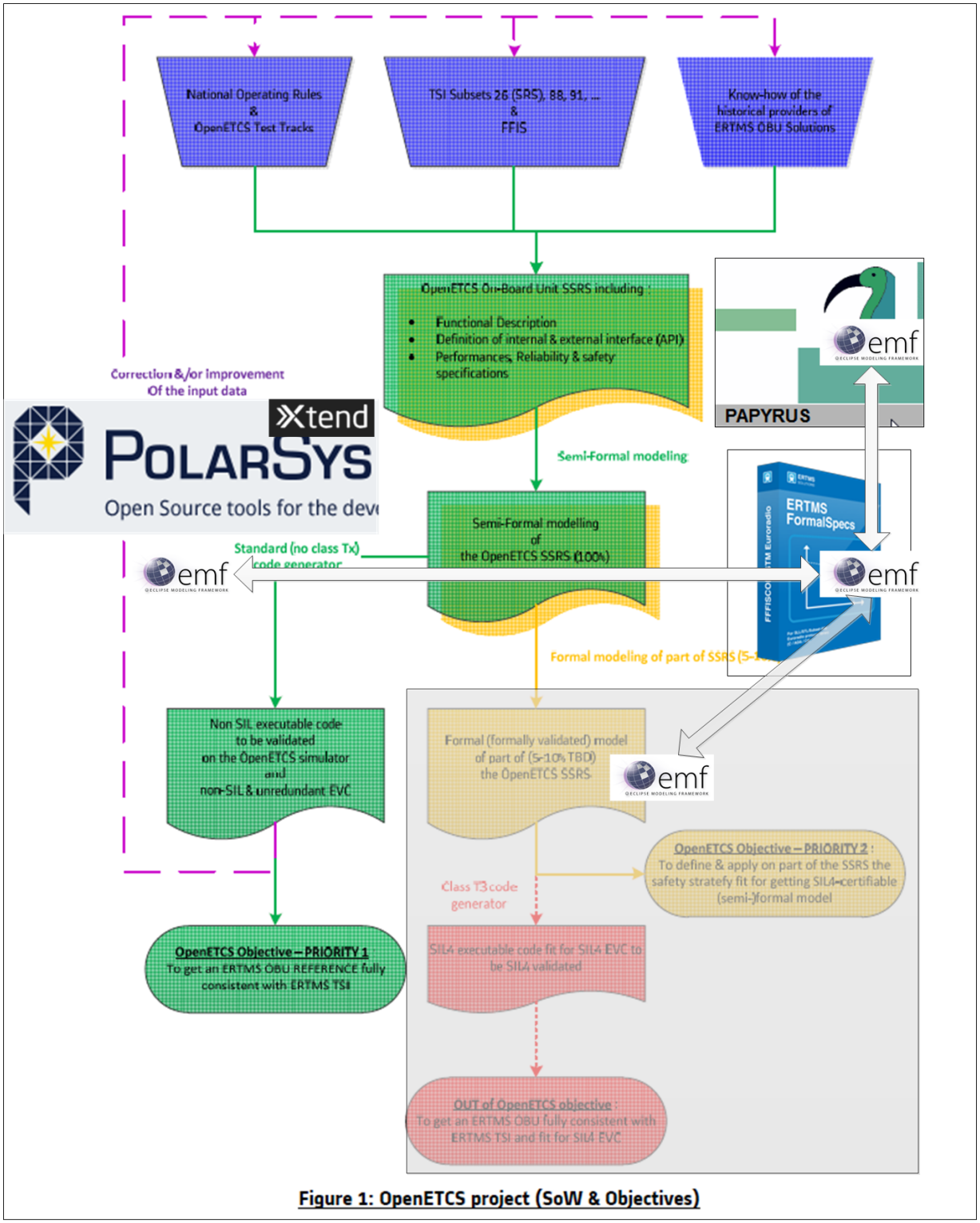


Figure B1. SysML, ErtmsFormalSpec and Eclipse/Polarsys Proposal

SysML/Papyrus is completely based on Eclipse.

ERTMSFormalSpecs supports today an EMF interface, enabling Eclipse-based tools to reuse the existing ERTMSFormalSpecs model.

Eclipse/Polarsys tools are also based on Eclipse, raison no integration concerns.

B.4 Benefits versus OpenETCS requirements

Todo: Discuss the benefits in regards of OpenETCS requirements and expected results.

The benefits of the SysML/Papyrus/ERTMSFormalSpecs/Eclipse/Polarsys proposal are the following:

- As of today, already 44% of Subset-026 requirements modelled. This proposal enables the OpenETCS project to start with a headstart, instead of nothing.
- ERTMSFormalSpecs is the only semi-formal candidate to have built-in braking curves modellings and visualization, verified with the ERA model
- ERTMSFormalSpecs has very strong support for traceability to the Subset-26, and to the Subset-076 for test cases
- ERTMSFormalSpecs has its domain-specific language, which is productive for this field, thanks to its expressivity, illustrated by the primitives developed for braking curves and scalable, as is demonstrated by the large fraction of the Subset26 which has been represented so far
- Fully open-source (ERTMSFormalSpecs under EUPL license, others open source)
- ERTMSFormalSpecs model can be transformed automatically to SCADE model (confirmed by ESTEREL Technologies in Munich meeting), allowing to choose SCADE as a code generation backend in case Eclipse/Polarsys would not meet the project requirements
- The three elements of the toolchain (Papyrus, ERTMSFormalSpecs and Eclipse) boast an active community of users and are supported by open-source business cases

B.5 Shortcommings versus OpenETCS requirements

Todo: Discuss the shortcommings in regards of OpenETCS requirements and expected results.

The shortcomings of the SysML/Papyrus/ERTMSFormalSpecs/Eclipse/Polarsys proposal are the following:

- ERTMSFormalSpecs has a perfectible look and feel and lacks graphical rendering of the architecture. *This shortcoming might be addressed with the integration of SysML as a language for the higher level architecture.*

- As of today, the code generation in Eclipse/Polarsys, transforming the ERTMSFormalSpecs in generated source code, is not yet available, and must be developed during task 3.8 of the project. *This shortcoming can be mitigated by integrating the scade code generator as a itermediates solution.*

B.6 On going work for openETCS project

Todo: Which are the elements to clarify, to specify or to develop, in order the approach suit the openETCS process ?

How can we evaluate and plan this work ?

which skills is needed ?

The following elements should be further developed during the OpenETCS project, to alleviate the shortcomings listed above:

1. Develop conceptual and technical strategies to integrate SysML with ERTMSFormalSpecs model, going beyond the state-of-the-art. The state-of-the-art in integrating SysML models and industrial (semi-)formal languages being SCADESystem, in which only the module, interfaces and dataflows are connected to the lower-level model.
2. Further develop ERTMSFormalSpecs model to cover 100% of Subset-026, 100% of Subset-076, and to fully test the model within ERTMSFormalSpecs.
3. Either implement a full eclipse-based version of the ERTMSFormalSpecs workbench for model development, traceability and testing, or improve the ERTMSFormalSpecs user interface to be fully productive for T3.5 and T3.6 tasks
4. Develop code-generation strategies during T3.8 tasks

Among theses tasks, Task 2 and 4 fit inside the WP3 existing tasks, and do not need additional skills than the ones present in the OpenETCS consortium as of today. Task 1 is also seriously represented in the consortium, in which a lot of SysML experience is present.

Task 3 (porting the ERTMSFormalSpecs workbench to Eclipse) is a task that falls in the scope of WP7, and for which Eclipse development and integration skills are required.

B.7 Conclusion and other comments

As a conclusion, the SysML/Papyrus/ERTMSFormalSpecs/Eclipse/Polarsys proposal has as 3 key strengths 1/ to be fully available today, 2/ to already model 44% of Subset-026, and 3/ to be fully open-source.

Moreover, it proposes a realistic technical foundation to achieve the WP3 and projects top-priority objectives with the skills and resources available in the project.

Appendix C: SysML and ClassicalB

Todo: Description of the approach by Alexander Stante.

This section describes the approach of combining SysML modeling with the B Method. The technical realization is shown in Figure C1. Modeling starts in SysML based on the tool Papyrus. At the current stage of the project, it is undefined which SysML elements and diagrams will be used. This must be well defined to ensure that a transformation from SysML to Classical B is semantically correct. To ensure this, the approach suggest the use of the *Object Constraint Language (OCL)* or the *Eclipse Validation Framework* for validating the SysML model according to defined modeling guidelines. Such guidelines may restrict the usage of certain modeling elements and may enforce certain naming conventions.

The validated SysML model will be transformed to a Classical B model with a model to text transformation language (e.g. Xtend2). That B model is considered as read-only and is only allowed to be further refined. From that point, the existing Atelier B toolchain will be used for refining the model until reaching the implementation model. Provers support the V&V activities and the open source code generator c4g is capable of generating C code.

The proposed toolchain is intended to be a first version. The yellow boxes in Figure C1 indicate non existing software artifacts that have to be developed within the openETCS project. Furthermore, it is desirable to move parts from the Atelier B tool into Eclipse.

In Table C1 the list of tools used in the proposed toolchain is given.

C.1 Description of the approach for OpenETCS design process

Todo: How the proposed approach covers all "blue" design process (see 1) ?

C.2 Integration of the approach with SysML/Papyrus

Todo: How the proposed approach can be integrated with the SysML/ Papyrus selected for the high level of design process ?

Tool	License	Link
Bart (B Automatic Refinement Tool)	GPL	`http://sf.net/projects/bartrefiner/`
Atelier B GUI	GPL	`http://sf.net/projects/atelierbgui/`
Bcomp (B Compiler)	GPL	`http://sf.net/projects/bcomp/`
C4b (C Code Generator)	GPL	`http://sf.net/projects/c4b/`

Table C1. Tools used in the Classical B toolchain

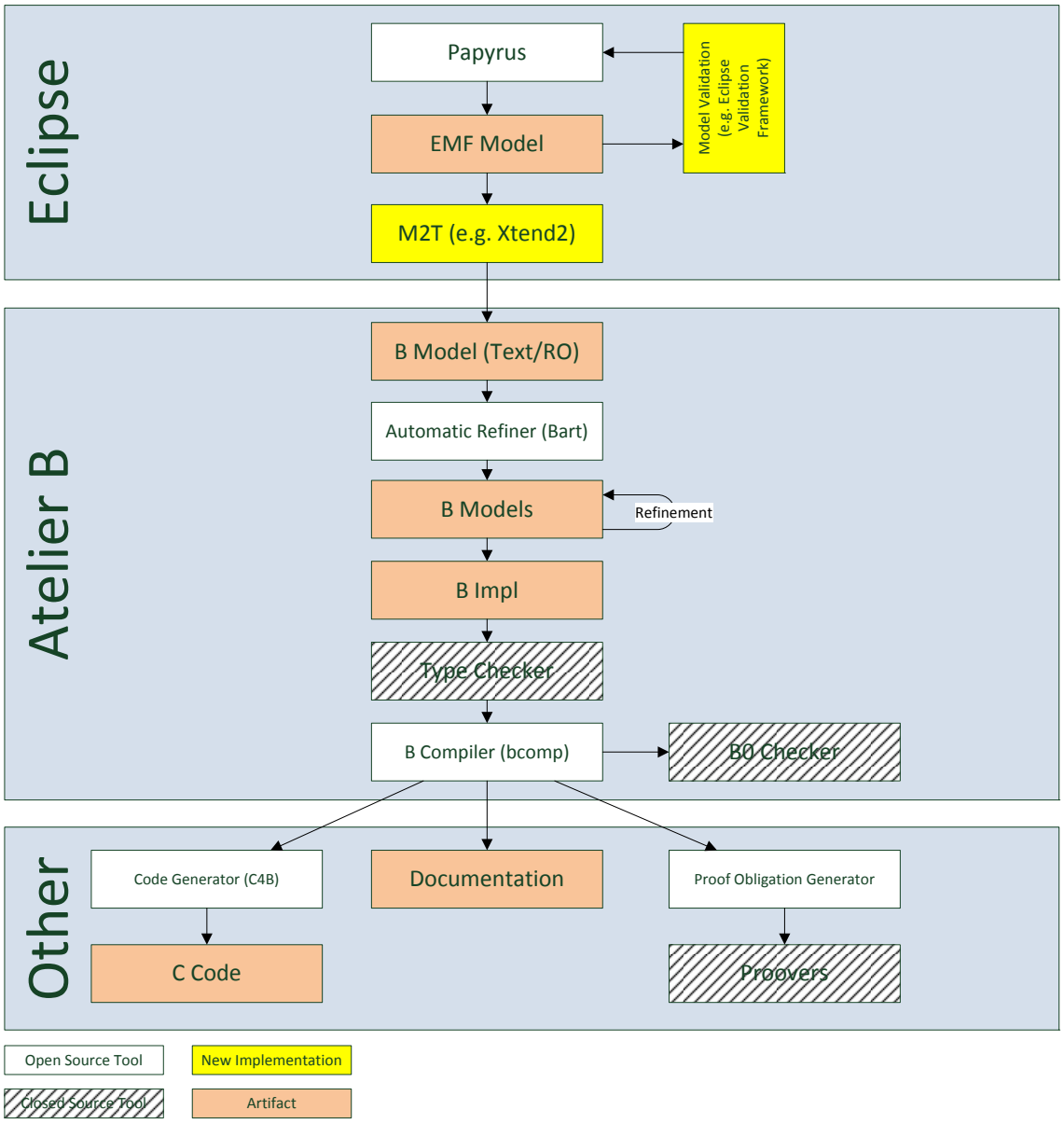


Figure C1. Technical overview of the Papyrus and Classical-B toolchain

C.3 Integration of the approach with Eclipse

Todo: How the proposed approach can be integrated with the Eclipse, selected as platform for OpenETCS tool chain ?

C.4 Benefits versus OpenETCS requirements

Todo: Discuss the benefits in regards of OpenETCS requirements and expected results.

C.5 Shortcommings versus OpenETCS requirements

Todo: Discuss the shortcommings in regards of OpenETCS requirements and expected results.

C.6 On going work for openETCS project

Todo: Which are the elements to clarify, to specify or to develop, in order the approach suit the openETCS process ?

How can we evaluate and plan this work ?

which skills is needed ?

C.7 Conclusion and other comments