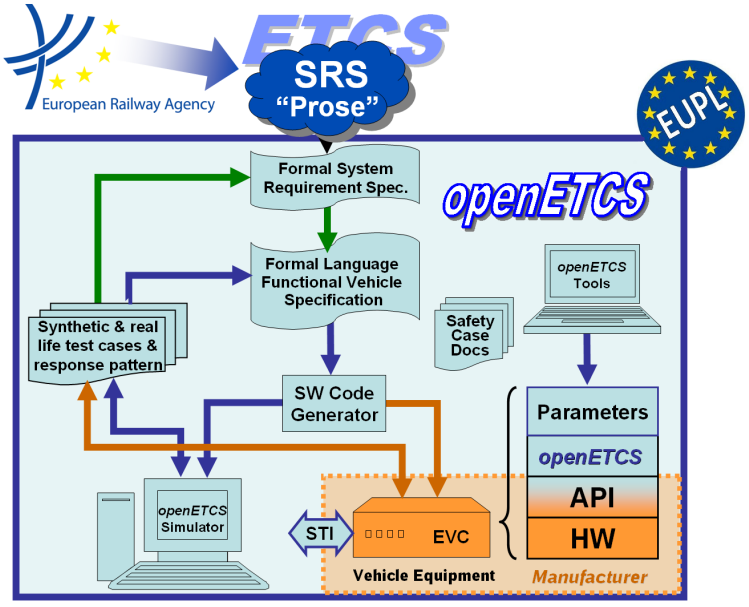
ITEA2 Project
Call 6 11025
2012 – 2015

Work Package 7, Task 1: “Primary Toolchain”

Report on the Final Choice of 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, Task 1: "Primary Toolchain" **OETCS/WP7/D7.1 – 00/05**
July 2013

Report on the Final Choice of 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 analyze and recommend, means, tools and platform to develop the primary toolchain 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
00.04	31/07/2013	M.Jastram	Significant work in §§1, 2, 3 and 4 before review.
00.05	13/08/2013	M.Petit-Doche	Github issues taken into account: 80, 84, 87, 88, 89, 90(partial), 96 107

1 Introduction

The aim of this document is to report the results of the task T7.1 of WP7: "Primary Toolchain Analyses and Recommendations".

1.1 Executive Summary

The task of WP7 is to produce an integrated chain of software tools. To simplify the selection process, these have been separated into a primary and secondary toolchain in the WP7 Description of Work (DoW) [1]. This document is concerned with the selection of tools for the primary toolchain. As the tools shall be integrated, the selection of an integration platform was required as well.

Eclipse was selected as tool platform, and the decision was unanimous 4.1.

Three toolchains have been proposed, which will be referred to with the following shortcuts for brevity:

SCADE. This toolchain is based on SCADE in connection with Papyrus, for which an integration already exists.

EFS. This toolchain is based on ERTMS Formal Specs (EFS), in connection with Papyrus, for which an integration has to be developed. It will take advantage of elements from the Eclipse Ecosystem (and from TOPCASED, wherever possible). The methods for integrating the various models have to be developed, as well as the gluing code that will hold everything together.

B. This toolchain is based on B in connection with Papyrus, for which an integration has to be developed.

SCADE is the perfect candidate for a backup plan: While offering pretty much everything the project needs, it has a major fault: SCADE is not open source. At the same time, SCADE is a backup that could be activated very quickly. It is preferable to model successfully with a closed-source solution than to fail with an open one. But of course, modeling successfully with an open source solution is what we should strive for.

As all three solutions use Papyrus as the first modeling tool, activities should focus on narrowing down which elements of Papyrus should be used. Once this is clear, WP3 can get started with their modeling activities.

Some effort needs to be spent on defining the interfaces between the various tool elements. How this could look like has been demonstrated nicely by the B-team, as visualized in Figure C1. We encourage the EFS-team to do the same.

The interfaces of the three toolchains should be aligned as much as possible. Doing so will allow to switch tool components between the three toolchains with comparatively little effort, if the needs arises.

Once the toolchains are defined in detail, a pilot model will be created using the B and EFS toolchains. This pilot will be the foundation for deciding on one single toolchain, by the end of 2013 the latest.

Section 4 contains specific decisions intended to concisely guide activities until the end of 2013.

1.2 T7.1 Objective

The goal of WP7 is to provide other openETCS work packages with tooling for their activities. Tools have been separated into primary and secondary tooling. Task T7.1 is concerned with the identification of the best primary tooling, considering all constraints of the project. Selecting tools also entails selecting modeling languages and an integration platform.

1.3 Scope of Task T7.1

The scope of the primary tooling has been defined in the WP7 Description of Work [1]. 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." (D2.3, 4.4.3). "The means of description of the semi-formal model shall be understandable by domain experts, providing graphical description" (D2.3, 4.4.4).

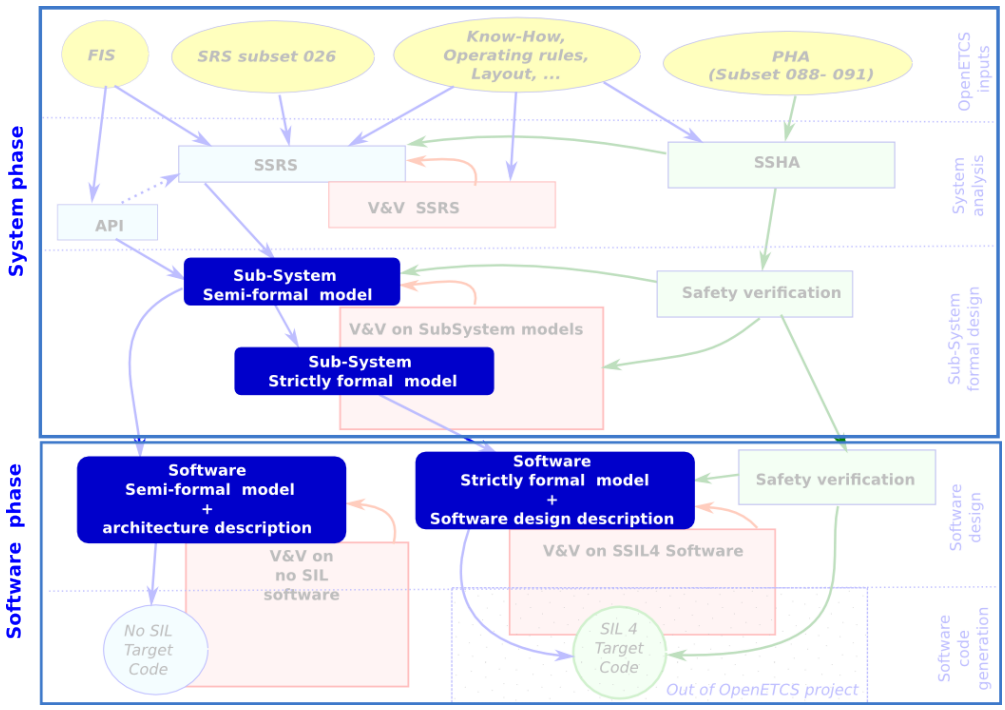


Figure 1. Main OpenETCS process. The models that are covered by primary tooling are shown in blue.

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." (D2.3, 4.4.2). "To facilitate safety activities, the safety relevant function should be as much as possible insulated from non safety relevant functions." (D2.3, 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." (D2.3, 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." (D2.3, 4.5.3).

1.3.1 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 toolchain that may affect the secondary toolchain 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 [2] 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 semi-formal model to describe the functional architecture of the on-board unit." (D2.3, 4.3.4 [2]).

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 standardized interfaces." (D2.3, 4.6.2).

1.4 Organization of this Report

This report is organized as follows:

[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).

Introduction (Section 1). An executive summary, as well as an overview of the WP7 Task T7.1 activities. It shows how T7.1 fits into openETCS in general and WP7 in particular. It also describes the scope of the results described in this deliverable, making sure the boundary to the secondary toolchain is properly defined.

Results on Means and Tools (Section 2). A description of the three proposed toolchains. The selection process is described in detail.

1.5 Glossary

API Application Programming Interface

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

DSL Domain Specific Language

EMF Eclipse Modeling Framework

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

SWOT Strengths, Weaknesses, Opportunities, Threats

V&V Verification & Validation

2 Results on means and tools for primary toolchain

2.1 Proposed Toolchains

This chapter describes the proposed toolchains and the selection process for finding them.

There were three toolchain 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 toolchain. 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 modeled with this toolchain. The biggest advantage of this approach is that it is open source, and that a significant portion of Subset 26 has already modeled 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

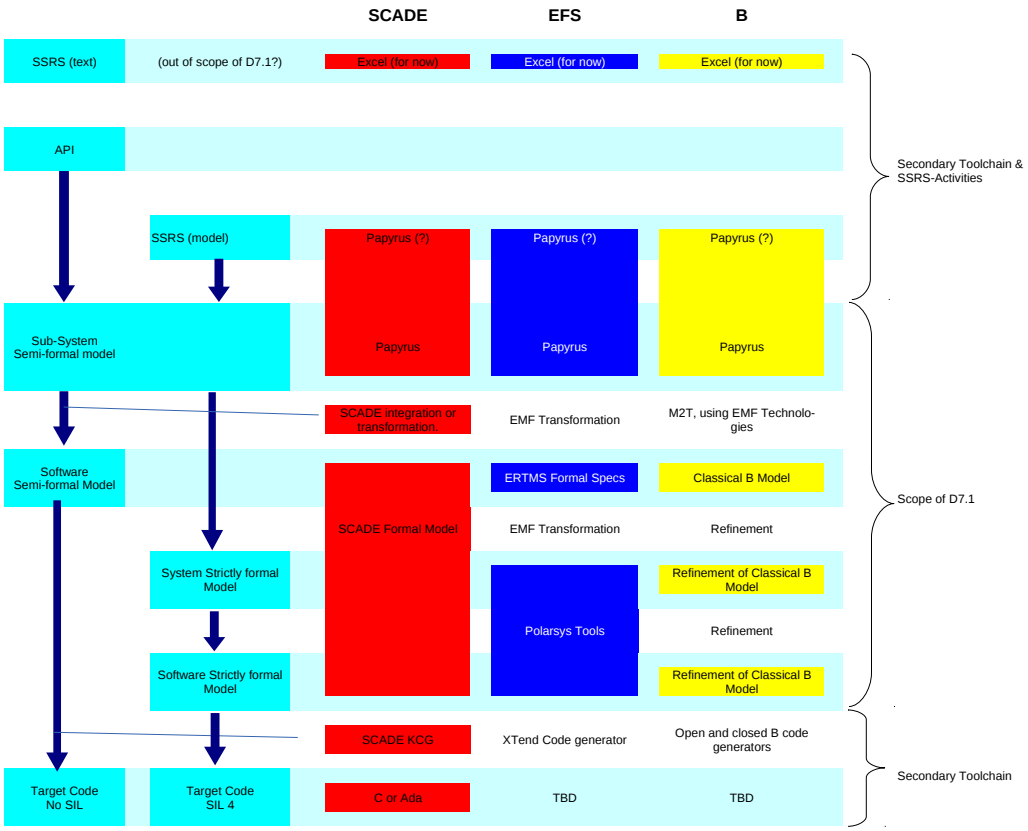


Figure 2. Proposed Toolchains

that it is not clear how laborious the tailoring of the Eclipse-based formal modeling tools would be.

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 toolchain 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 Selection Process

The selection process consisted of the six steps shown here, with more detail being provided below.

Benchmarking. (Section 2.2.1) Project partners were asked to step up and perform a benchmark of the tool of their choice. The proposals were checked against D2.1 (State of the art) to make sure that no important tools were missed (or if so, why). Not all benchmarks were completed, CORE, Why3 and UPPAAL dropped out. Further, after completing the benchmark, GOPPR, GNATprove and Petri Nets decided to run, due to the evaluation results.

Assessment (Section 2.2.2). Each Benchmark was assessed by its author and by two additional assessors (in the case of Petri Nets and GNATprove, there was just one assessor). The assessment quantified evaluation criteria, resulting in a report consisting of hundreds of scores for each completed benchmark.

Decision Meeting(Section 2.2.3). A decision meeting took place to eliminate tools and to start looking at possible tool configuration that would form the primary toolchain. During this meeting, Enterprise Architect (SysML) was also eliminated, in favor of the open source alternative Papyrus. Further, Papyrus/SCADE was identified as one promising toolchain. Further, Papyrus had been selected as the tool for the Sub System Semi-formal model.

Composition of Toolchains (Section 2.2.4). With the number of tools reduced significantly and one selected tool (Papyrus), team members were invited to propose concrete toolchains covering the primary toolchain. This resulted in two more proposals, in addition to Papyrus/SCADE.

Documenting the Decision (Section 2.2.5). This deliverable documents the three toolchains, as well as their respective strength and weaknesses. As has been determined a long time ago[2], we didn't expect a clear single "best" solution. The document will steer the upcoming activities to prevent redundant or unnecessary work.

Final Choice of Toolchain. Which of the three competing toolchains will eventually be used for openETCS shall be decided six month after D7.1 the latest (as documented in the tool selection process). Note that this may not be one of the three suggested, but could also be a composition of elements of the three toolchains.

[2]Tool Selection Process: https://github.com/openETCS/model-evaluation/wiki/Benchmark–to-evaluate-model-and-tools#tool-selection-process

2.2.1 Benchmarking

The following thirteen tools have been selected for benchmarking:

SCADE	ERTMSFormalSpecs
SysML with Papyrus	SysML with Enterprise Architect
Classical B with Atelier B	Event-B with Rodin
System C	Petri Nets‡
GOPRR‡	GNATprove‡
UPPAAL†	Why3†
CORE†	

† The evaluation of three tools was stopped prematurely. They are mentioned in [3], but are not evaluated. They dropped out for the following reasons:

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 toolchain:

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

GNATprove According the results, GNATprove has been proposed to joined the evaluation of secondary tools (task T7.2).

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

2.2.2 Assessment

Each Benchmark was assessed by its author and by two additional assessors (in the case of Petri Nets and GNATprove, there was just one assessor). This has been documented in [3], which contains the “raw data”, as well as some rudimentary aggregation of the data and some analysis.

The criteria in [3] were derived from WP2 requirements and quantified on scale from 0 to 3. The results were recorded as the sum of the author and the two assessors, resulting in a score from 0 to 9 for each criteria. The benchmarks with only one assessor have been adjusted by interpolating the score, shown in parentheses.

To give an idea of the results, Table 1 shows the aggregated result of the benchmark for process phases. These results must be taken with a grain of salt: This aggregation simply averages all

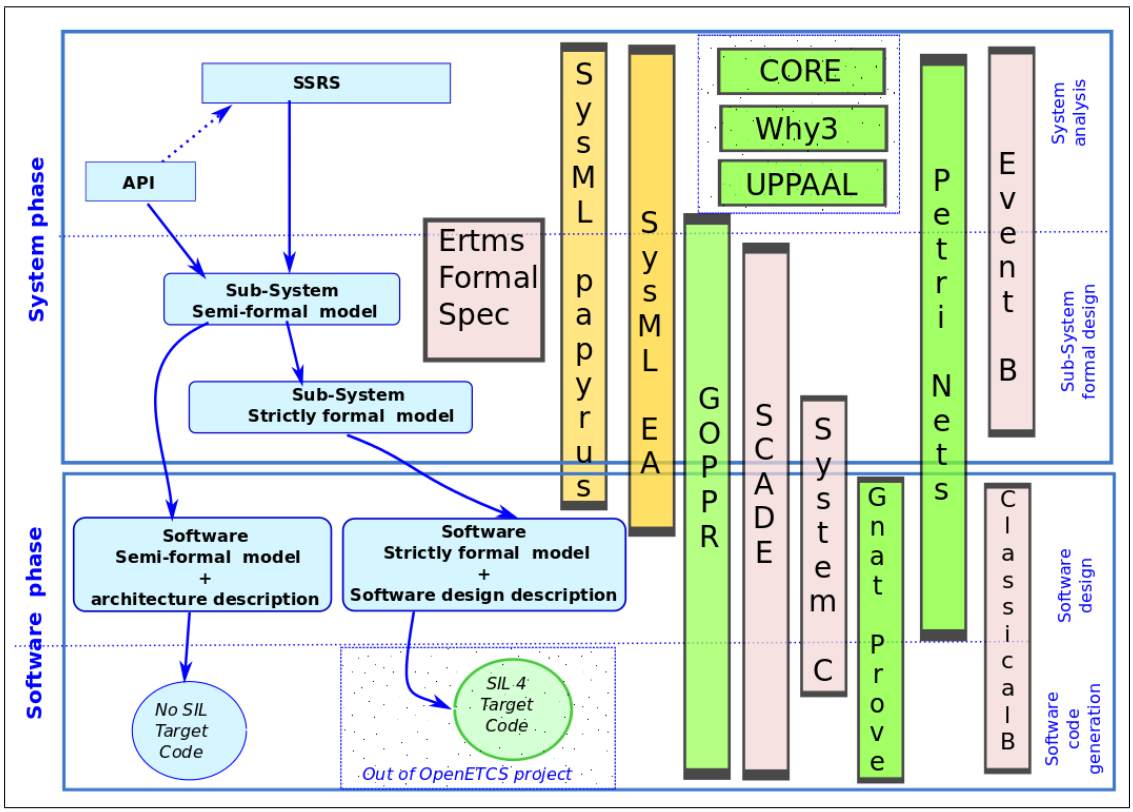


Figure 3. Results of candidates

relevant scores, thereby giving all criteria equal weight. This is an unrealistic (and dangerous) assumption. In fact in the case of the criteria "open source" it created a real problem. As Open Proof is the foundation of openETCS, using a closed-source tool with no open alternative should be a show stopper. But this does not show in the results of [3].

Further, not all criteria could be quantified easily, resulting in surprising and suspicious results. For instance, the assessments of Papyrus and Enterprise Architect sometimes differed drastically, even in areas, where the notation (and not the tool) was concerned. This was not expected, as both tools support the same notation. This means that the results must be inspected carefully, before drawing conclusions.

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with EA	SCADE	Event-B	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)**

Table 1. Use of the approaches during process phases

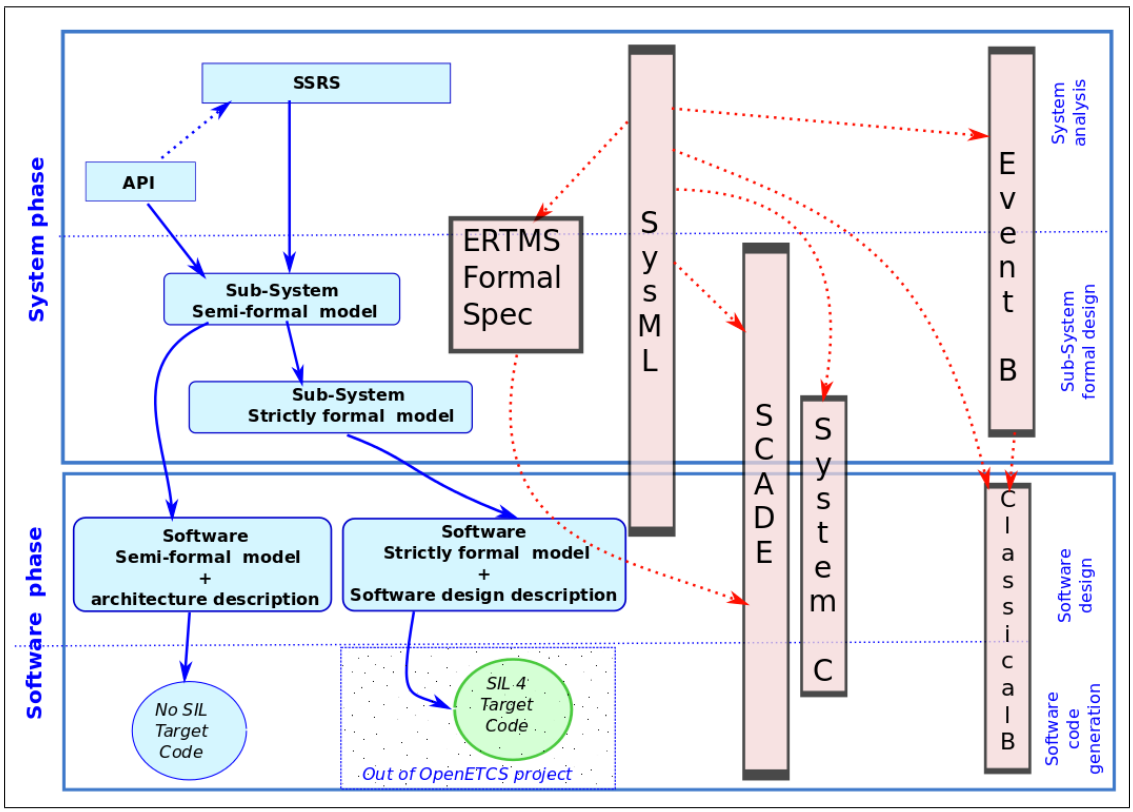


Figure 4. Short list

2.2.3 Decision Meeting

On July 4th, a WP7 workshop took place with the objective of choosing suitable tools for the toolchain. To ease the process, the results from [3] were condensed into Figure 3. The vertical stripes represent the individual tools. The length and position of the stripe is determined by Table 1: The bar only covers area with a score of 6 or higher.

2.2.3.1 SysML

Using the available data as the foundation, the objective was to narrow the choices down as much as possible. One obvious redundancy was the availability of two SysML tools, Papyrus and Enterprise Architect. After discussion of pros and cons of each, the partners agreed that Papyrus covers better the objectives of the project, especially the open-source requirement.

Further, by vote, all the partners agree on the use Papyrus/SysML to cover the higher level of the OpenETCS V-cycle.

2.2.3.2 Short List

Due to the evaluation and discussion during the decision meeting, the partners agreed on a short-list of six approaches. These are shown in red in Figure 4. and include the following:

SCADE	ERTMSFormalSpecs
SysML with Papyrus	System C
Classical B with Atelier B	Event-B with Rodin

2.2.4 Composition of Toolchains

One toolchain was already suggested during the Paris meeting, namely Papyrus and SCADE. Nevertheless, partners had the chance to propose additional toolchains using the tools from the shortlist. It was necessary to find alternatives, as one severe weakness of the Papyrus/SCADE solution had already been pointed out in Paris: By not being open source, this solution would miss one of the main objectives of openETCS.

Two more toolchains have been proposed, resulting in three toolchains in total. These have been documented by their respective owners in the appendix.

2.2.5 Documenting the Decision

What has been documented so far is a promising foundation for an openETCS primary toolchain. However, it is not clear on how to proceed from here, as there are multiple competing avenues. Resources in the project are thin and should be employed wisely. Therefore, Section 4 contains an elaborate analysis with clear decisions that will guide the activities of the next six month, ensuring that we will choose the best option for openETCS while minimizing risk.

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.

3.1 Initial list of candidates

- Eclipse
- TOPCASED/Polarsys
- CAESAR RTP
- Mono/.NET
- SCADE

After a first round, Mono/.NET and SCADE were discard because they do not comply to our tool platform definition. CESAR RTP 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 life cycle 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 [4], it was not evaluated as a whole. Merely the Papyrus 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

As anticipated there is not one single tool choice. This has the advantage of reducing risk (if a tool does not work out), but the disadvantage of potentially wasting resources and being unfocused. The objective of this chapter is to summarize the decision and to propose a work plan that leverages the advantage of having multiple tools, while reducing the risks resulting from this. However, the choice of the tool platform was easy and unanimous.

4.1 Decision on the tool platform

By vote in Paris on July 4th, 2013, all the partners agree on the use of **Eclipse** as tool platform. Which version of Eclipse will be used is up to the implementing team (WP7.3). The last release is Kepler 4.3, launched the 26 of June 2013.

4.2 SWOT Analysis

By looking a the strengths, weaknesses, opportunities and threats (SWOT) of each toolchain, we will get a qualitative impression on what we could gain with each solution, and what the risks are.

4.2.1 SCADE SWOT Analysis

4.2.1.1 Strengths (SCADE)

The biggest strength of SCADE by far is that it works “out of the box”, barely any tailoring is required. SCADE has been developed for systems engineering, and this is exactly the right field of application. Further, a Papyrus integration is already available, so little work has to be done here. Another strength is the fact that some of the secondary tool activities are covered by SCADE as well.

4.2.1.2 Weaknesses (SCADE)

The biggest weakness of SCADE is a show stopper: SCADE is not open source, and as no open source alternative exists, openETCS would miss its open proofs objective.

4.2.1.3 Opportunities (SCADE)

Using SCADE would doubtless increase the chances of success of the modeling activities. Thus, SCADE is an excellent backup plan. By nominating SCADE as a backup, we would ensure damage control: A successful model, even if not created with open tools is preferable to no model at all.

There is another opportunity: by dangling the chance to adapt SCADE and potentially opening a large market segment, Esterel (manufacturer of SCADE) may decide to open SCADE — at least enough for our purposes.

4.2.1.4 Threats (SCADE)

There is the real threat that Esterel is encouraging us to adapt SCADE under the premise of opening parts of SCADE to be compliant with open proofs. If such discussions would breakdown, after modeling has already started, we'd have a problem.

4.2.2 ERTMS Formal Specs SWOT Analysis

4.2.2.1 Strengths (EFS)

EFS is open source. Even better, a significant portion of the ETCS specification has already been modeled using the tool. The creator of the tool (ERTMS Formal Specs) is available in the project and has project resources for WP7, which should result in fast turnaround during development.

It has already been proven that all features of the ETCS specification can be modeled.

EFS has been developed as a commercial tool and is based on real-world needs in the rail industry.

4.2.2.2 Weaknesses (EFS)

EFS has originally been written using the .NET platform. Even though a prototype based on Eclipse EMF exists, a significant amount of work is required to make it truly user friendly. However, it should be possible to work during a transition phase with the old tool.

As the notation of EFS is more of a domain-specific language (DSL), A debate has been going on whether EFS is "formal enough". This is not a big problem, as the model could be extended with fully-formal models in relevant places. A bigger question is how the integration of the various models would be realized. There won't be a clear answer on that, until we try it out.

Last, the "lower part" of the toolchain is citing technologies that need a significant investment of energy before they can be used. For instance, Xtend is a programming language, which is not doing anything domain-specific.

4.2.2.3 Opportunities (EFS)

The main opportunity is to safe time and resources, as a significant portion of the ETCS specification has already been modeled. Another is the certainty that a commercial partner will be engaged in the ongoing activities, even after the end of openETCS.

4.2.2.4 Threats (EFS)

The continuation of the toolchain, as shown in Figure B1, leaves a lot of questions open. Compare that to the much more concise description of the B-approach in Figure C1. Due to the ambiguity there is a significant risk that the model integration won't work as shown.

4.2.3 B SWOT Analysis

4.2.3.1 Strengths (B)

B is proven in the rail industry, where it has been deployed successfully. There is also a strong body of academic research that can be taken advantage of.

There is a decent amount of tooling already available, both open source and commercial. However, an almost complete open toolchain has been suggested in Table C1, with the Atelier B type checker being the only exception (see Weaknesses below).

There is B-related expertise in the consortium, ensuring that questions can be answered and that there is a commercial incentive.

4.2.3.2 Weaknesses (B)

Atelier B not Eclipse-based, and Appendix C points out this shortcoming. On the positive side, it is available on all relevant platforms (Windows, Mac, Linux).

There is also the Atelier B type checker, which is not open source. Finding an open replacement for this relatively small component would be one mandatory activity.

While B is well-suited for modeling state-based systems, it is not clear how continuous modeling (e.g. breaking curves) would be realized.

4.2.3.3 Opportunities (B)

B could be the sweet spot between using a commercially proven approach (like SCADE), while still residing in the open source realm (like TOPCASED). And as there are more options available, both open and closed, the risk is lower.

4.2.3.4 Threats (B)

Acceptance may be the biggest problem, as B is a “hard core” formal notation, which is considered hard to read. We must be prepared to train the users and ensure that they accept it beforehand.

4.3 Decisions

In the following, we will document a number of decisions that follow from our analysis and that will guide the activities until the end of 2013.

4.3.1 SCADE

The analysis shows that we have two promising open proposals (EFS, B), and a third non-open proposal, that is extremely powerful (SCADE). Considering the constraints of the project, it is clear that SCADE should only be employed if everything else fails. It is a “Plan B” that we should be grateful for, as it may allow us to make openETCS at least a partial success, in case of a larger tool crisis (of whatever nature). Therefore, we decide:

Decision 1

The SCADE toolchain will be the “Plan B” toolchain, to be employed if “all else fails”.

Decision 2

Therefore, activities on the SCADE toolchain will be suspended for the time being.

If we are in a real crisis (this will hopefully never happen), we'll have to put relatively little effort into migrating whatever has been done to the SCADE toolchain. This will be particularly true if we carefully define the scope of the Papyrus component.

The tools covered here must allow working with the four models shown in Figure 1.

4.3.2 Papyrus/SysML

All proposed toolchains will use SysML with Papyrus. To keep things as flexible as possible, it would be desirable to use SysML in exactly the same way for all approaches. The B team suggested to create modeling guidelines, and to build a validator for validating the SysML model accordingly.

Decision 3

All three teams will work together to create modeling guidelines that will apply to all three toolchains. If this turns out not to be possible, they should create a superset guideline, with extensions for the three approaches.

4.3.3 Aligning EFS and B toolchains

For better or worse, there is no clear winner between the EFS and B toolchains. We will know more when the rubber hits the road — when we start modeling. Therefore, modeling activities should start as soon as possible with a well-defined case study.

Decision 4

As the toolchains are being built, modeling will start immediately on a small, well-defined case study, which will represent a subset of the spec. The model will cover the tool from top to bottom, thereby demonstrating that the toolchain actually works as advertised.

It is conceivable to continue with the radio subsystem that has been used for the benchmarking, but the modeling expert may suggest a better case study.

It is quite conceivable that no clear winner emerges, but that individual components show unexpected strengths or weaknesses. Therefore, the toolchain must first define clear interfaces, similar to what the B team already hinted at in Figures C1 and C2. Again, there is no reason to not look for similarities between the two open toolchains and to align the interfaces, wherever it makes sense. For instance, if the Polarsys tools are not performing as expected, they may be replaceable by B tools. Therefore:

Decision 5

All toolchains shall clarify their architecture, keep it as modular as possible and align interfaces between them, wherever it makes sense.

4.4 Conclusion

While there was some discontent in the team of not having one single solution to move forward with, we simply have to turn this to our advantage. Having options helps us reduce risk, and by aligning the activities, we prevent our effort from dissipating. We believe that the documented decisions will act as high-level guidelines for achieving this goal.

Appendix A: SysML and SCADE

A.1 Description of the approach

Diagram A1 illustrates the most important components and operational relationships of a system and software modeling 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.

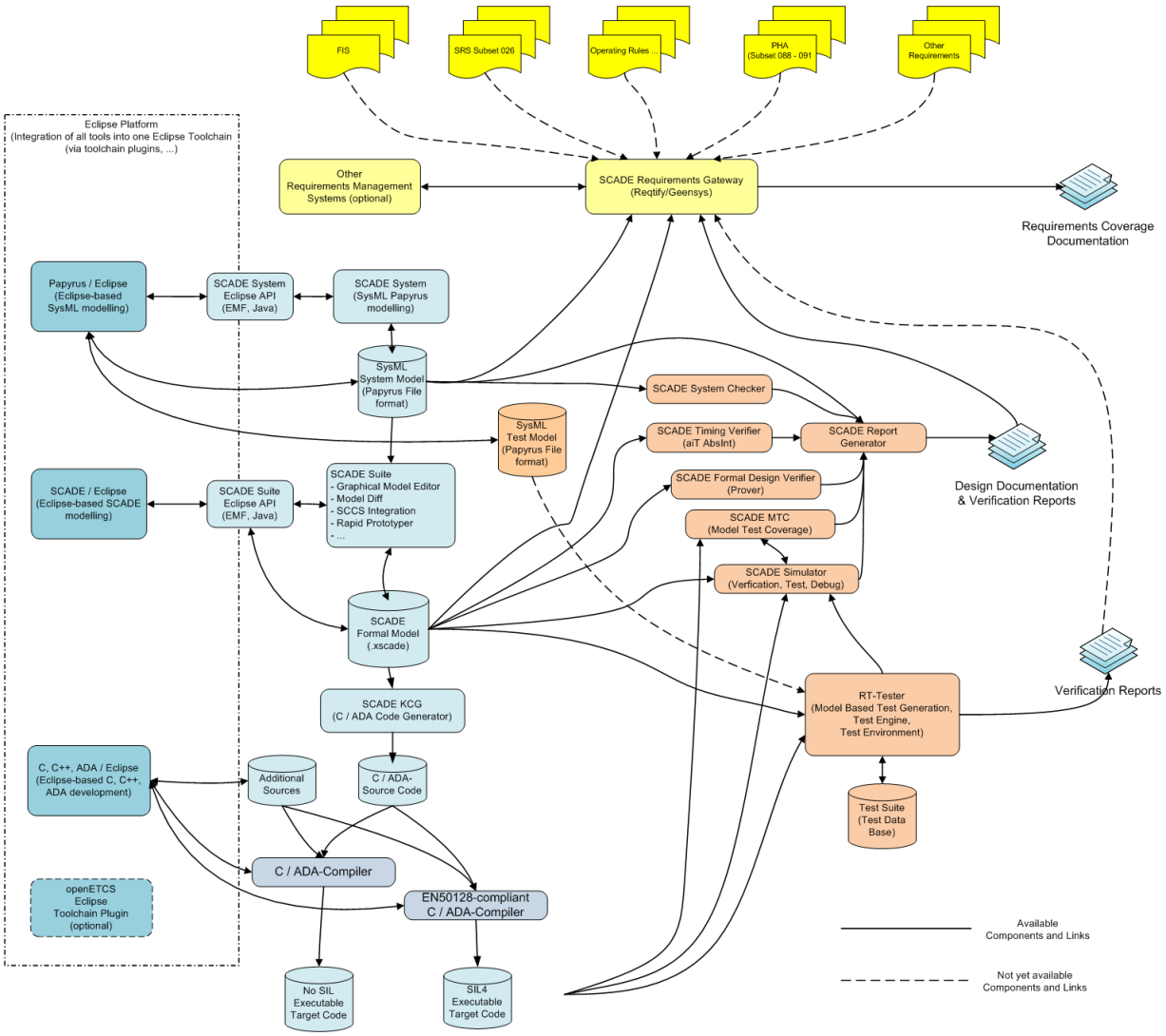


Figure A1. SysML SCADE Toolchain

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

- Requirements and requirements management components in yellow
- "Blue" openETCS design process (see figure 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.1.1 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 and verification documents, model and source code artifacts, 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.1.2 Semi-formal System and Subsystem Modeling 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 modeling 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 artifacts 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 Eclipse, 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, behavioral 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.1.3 Formal Modeling with SCADE Suite

SCADE Suite integrates all modeling, verification 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 modeling)
- SCADE Requirements Gateway Integration (Linking of model artifacts 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 modeling are editor and code generator. The others mentioned are mainly verification tools, but very useful and practically indispensable for agile development.

At least, to cover all elements of the "blue" design process in figure 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.1.4 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 verification 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 verification based on aiT from AbsInt.
- SCADE Report Generator: Generation of design reports for SCADE Suite and SCADE System model as 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.2 Description of the approach for OpenETCS design process

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

A.3 Integration of the approach with SysML/Papyrus

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 artifacts on system level should be modeled with SysML, that can not be transferred to native SCADE automatically.

A.4 Integration of the approach with Eclipse

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

SCADE System (SysML modeling) and SCADE Suite (SCADE modeling) 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 modeling 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 worthwhile 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.5 Benefits versus OpenETCS requirements

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

- seamless integration,
- completeness,
- maturity,
- qualification for safety critical development,
- productivity
- availability 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 modeling 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.6 Shortcomings versus OpenETCS requirements

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.7 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:

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.8 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 semi-formal modeling:

1. Use SysML/Papyrus for semi-formal modeling by utilizing only the SysML language artifacts, 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 artifacts 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 truly complete open source and open proof openETCS tool chain can be set up without causing unacceptable impact on the modeling work until it becomes mature enough for practical usage.

Appendix B: SysML, ERTMSFormalSpecs and Eclipse/Polarsys

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

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

SSRS box: Using Papyrus for modeling 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

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

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.

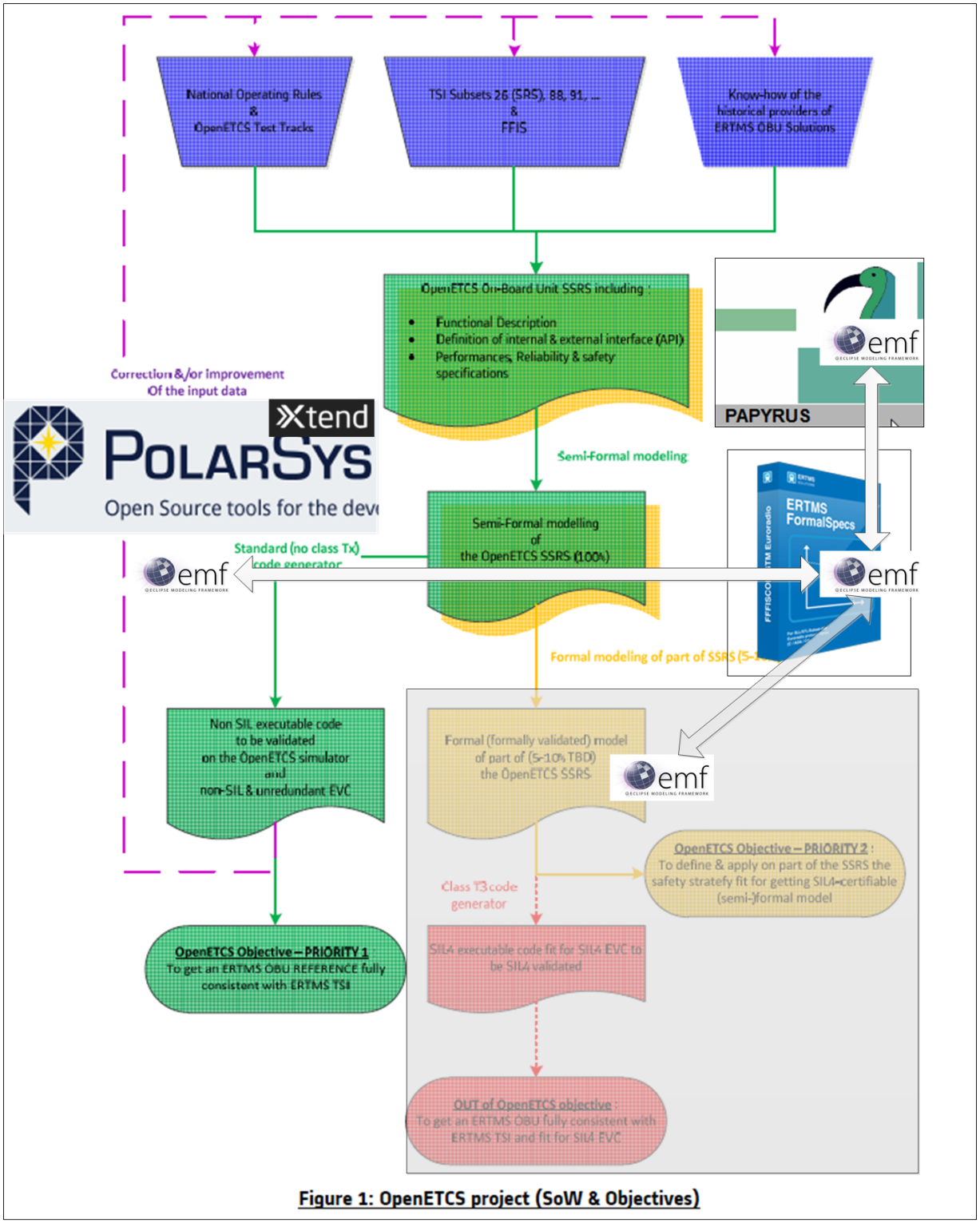


Figure B1. SysML, ERTMSFormalSpecs and Eclipse/Polarsys Proposal

B.4 Benefits versus OpenETCS requirements

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

- As of today, already 44% of Subset-026 requirements modeled. 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 modelings 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 Shortcomings versus OpenETCS requirements

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 intermediate solution.*

B.6 On going work for openETCS project

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 SCADE System, 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 Classical B

This section describes the approach of combining SysML modeling with Classical B. 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 validate the SysML model according to define modeling guidelines, the approach suggest the use of the *Object Constraint Language (OCL)* or the *Eclipse Validation Framework*. 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. Xtend). The generated Classical 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. Provers support the V&V activities and the open source code generator c4b 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. The GPLv3 licensed tools are stand-alone executables. It is not expected that GPLv3 licensed tool will lead to license issues with EUPL code, as these tools are only executed and not linked with other parts of the toolchain. The ComenC code generator is listed in the table, because the current version of Atelier-B ships with it instead of c4b. The source code of ComenC is available, however it is unclear under which license the source code is distributed. The Type Checker, B0 Checker, Prover and Proof Obligation Generator are currently only distributed under a proprietary license. However, a GPLv3 licensed Proof Obligation Generator is currently under development within in CERCLES-2 project.

C.1 Description of the approach for OpenETCS design process

Taking into account the results presented in Section 2, the proposed toolchain completely covers both, the system phase as well as the software phase (see Fig. 3). SysML with Papyrus would be used for modeling the SSRS on analysis level. The same tools will be used for the sub-system semi formal model, which will be created during the sub-system formal design phase. The SysML models will then be transformed to Classical B models, with the intention to perform further development steps according to the B method.

For both, the sub-system strictly formal model and the software semi-formal model it is proposed to use Classical B. The development of the B model will be done by further refining the read-only B models, which was generated from the SysML model. It has to be investigated where *exactly* the transition between SysML and Classical B will be done. In particular, it is currently unclear

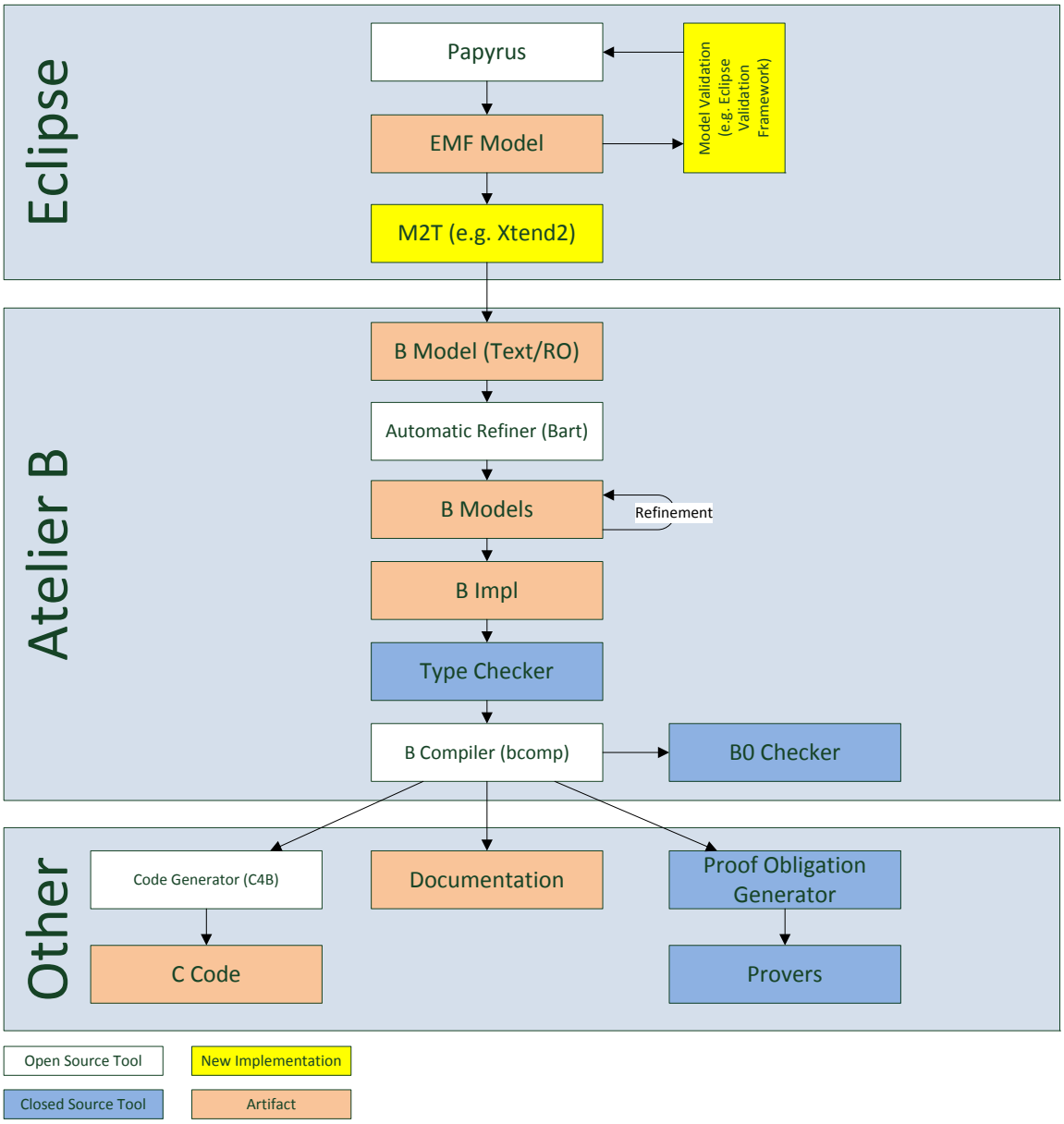


Figure C1. Technical overview of the Papyrus and Classical-B toolchain

Tool	License	Link
Bart (B Automatic Refinement Tool)	GPLv3	http://sf.net/projects/bartrefiner/
Atelier B GUI	GPLv3	http://sf.net/projects/atelierbgui/
Bcomp (B Compiler)	GPLv3	http://sf.net/projects/bcomp/
c4b (C Code Generator)	GPLv3	http://sf.net/projects/c4b/
ComenC (C Code Generator)	-	http://sf.net/projects/comenc/
Papyrus	EPL	http://www.eclipse.org/papyrus/
Xtend	EPL	http://www.eclipse.org/xtend/
Type Checker	proprietary	-
B0 Checker	proprietary	-
Prover	proprietary	-
Proof Obligation Generator	proprietary	-

Table C1. Tools used in the Classical B toolchain

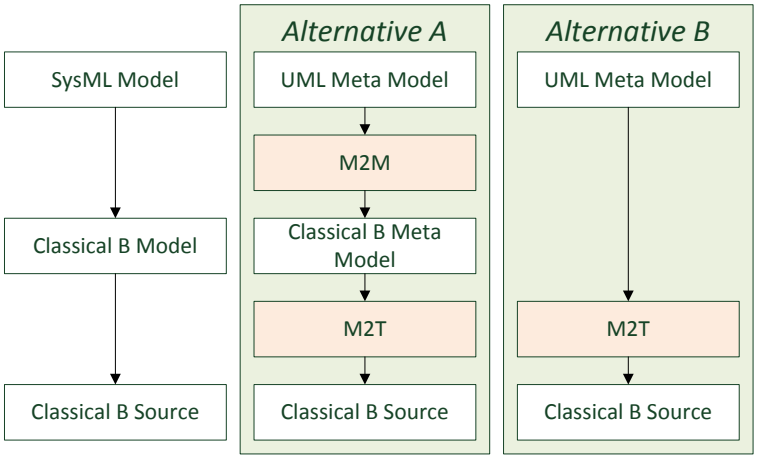


Figure C2. Alternatives in the transformation from SysML to Classical B

which SysML language constructs and diagrams will be used that can be transformed to Classical B. The structure described in SysML with *Block Definition Diagrams (BDD)* and *Internal Block Diagram (IBD)* are obviously prime candidates for a transformation to Classical B, but which behavior diagram will can be transformed is still under investigation.

The last part of the development process is the code generation and it is proposed to be performed with an existing code generator tool (c4b).

C.2 Integration of the approach with SysML/Papyrus

On the technical side, the integration will be performed by utilizing the EMF framework of Eclipse. With an appropriate transformation language (e.g. Xtend), the SysML model can be transformed to a textual representation in Classical B. Whether or not it is advantageous to perform the transformation with an intermediate Classical B meta-model and an a mode-to-model transformation (see Fig. C2) is under investigation.

The semantic integration of SysML and B method imposes a greater challenge. There already exist literature regarding the alignment of SysML and the B method. The work in [5] provides such an alignment focusing on V&V, which could be used as a base for further research, that considers in particular the openETCS requirements.

C.3 Integration of the approach with Eclipse

The integration of the proposed toolchain is mainly done as described in the previous section. After using SysML, the used tool is changed to Atelier B. This obviously does not correspond to a fully Eclipse based toolchain. However, this proposal only reflects an initial toolchain, which could be improved during the project and beyond. The parts of the toolchain which are user visible and realized with Atelier B in the first version, could be progressively moved to an Eclipse based solution. To what extend components from existing solutions (e.g. the Rodin platform) can be used and if there are enough resources available in the project, has to be investigated.

C.4 Benefits versus OpenETCS requirements

Both, SysML and the B Methods are accepted in the industry and have been successful used in the development of safety critical embedded systems. The B Method was used in projects like KVB by Alstom, SAET METEOR or in the driverless metro line 1 in Paris by Siemens Transportation Systems (see [6] for an extensive list).

Most parts of the toolchain are already available and distributed under an open source license. The transformation from SysML to B is the only missing software artifact, that has to be developed for the first version of the toolchain (see Fig. C1).

The project consortium includes partner with great expertise in both languages, SysML and B. By incorporating these expertise, there is a good chance that the proclaimed goals of openETCS will be achieved.

The modular architecture allows the exchange of certain tools with more powerful alternatives, even if they are closed-source. For example, there is a closed source code generator which allows the creation of ADA or C++ code. This code generator could be used as a drop-in replacement for the open-source c4b if necessary.

Furthermore, with Eclipse and Atelier-B versions for Windows, Mac OS and Linux, the most common operating systems are supported by the proposed toolchain.

C.5 Shortcomings versus OpenETCS requirements

Although the majority of the toolchain is open-source, there is still a small amount of tools which are closed source (see Fig. C1). During this project, these parts will be further investigated with the goal to be replaced by open-source equivalents.

A general weak point of the approach is the interface between Eclipse and Atelier-B. It would be favorable if the tool flow is only unidirectional. However, if during refinement steps in Classical B an bug in the initial B model is encountered, a correction in the original SysML model is necessary. This change triggers the generation of a new, initial B model. A bi-directional transformation could ease this issue, which would require a tighter integration of both tools. This is amongst other one reason why moving parts from Atelier-B to Eclipse should be further

investigated. This issue applies to all proposed toolchains as all change at a certain point the modeling tool and the used language.

The example from the previous paragraph also show that the modeler should preferable have knowledge of both, SysML and Classical B. The reason is that otherwise bugs found in the SysML model during the Classical B development phase have to be fixed by the modeler of the original SysML model, which may be inefficient. This issue should be also considered when the border between SysML and Classical B is defined.

C.6 On going work for openETCS project

One of the most important parts is to define a subset of SysML which can be transformed to Classical B and preserves it semantics. Probably the other toolchains also need a restricted usage of SysML, therefore a collaboration on this topic would be advantageous. Furthermore, it would be ideal if one subset could be defined, which is applicable to all proposed toolchains.

Another ongoing work is the precise definition of the border between SysML and Classical B. This work is also linked to the work described in the previous paragraph, as the border my have implication on diagrams used in SysML.

The transformation from SysML to Classical B must be planned and developed within this project. On the implementation part a desirable skill is the knowledge of transformation languages in Eclipse.

Experience has shown that for model transformations the source model should be checked according to guidelines. An obvious one is naming as the source model may allow character combinations that are prohibited in the destination model.

To show the capabilities and to identify any currently unforeseen barriers, an early prototype should be developed which illustrates all steps using an example use-case.

C.7 Conclusion and other comments

The proposed toolchain based on SysML and Classical B covers the openETCS design phases from system analysis, sub-system formal design, software design and software code generation.

Both, the proposed languages and tools have industry acceptance and were used in projects developing safety critical embedded systems.

With the exact definition of the border between SysML and Classical B, only the transformation between both languages has to be developed for an initial version of the toolchain. During the project, this initial version can be gradually improved to remove the last remaining closed-source (see Fig. C1) tools.

Appendix: References

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

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

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

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

[5] Erwan Bousse, David Mentré, Benoit Combemale, Benoit Baudry, and Katsuragi Takaya. Aligning SysML with the B Method to Provide V&V for Systems Engineering. In *Model-Driven Engineering, Verification, and Validation 2012 (MoDeVVa 2012)*, Innsbruck, Autriche, September 2012.

[6] ClearSy System Engineering. The b formal method: from research to teaching. Presentation held 16. June in Nantes, June 2008.