OETCS/WP7/O7.1.3_O7.1.7 – 00/02

openETCS

ITEA2 Project
Call 6 11025
2012 – 2015

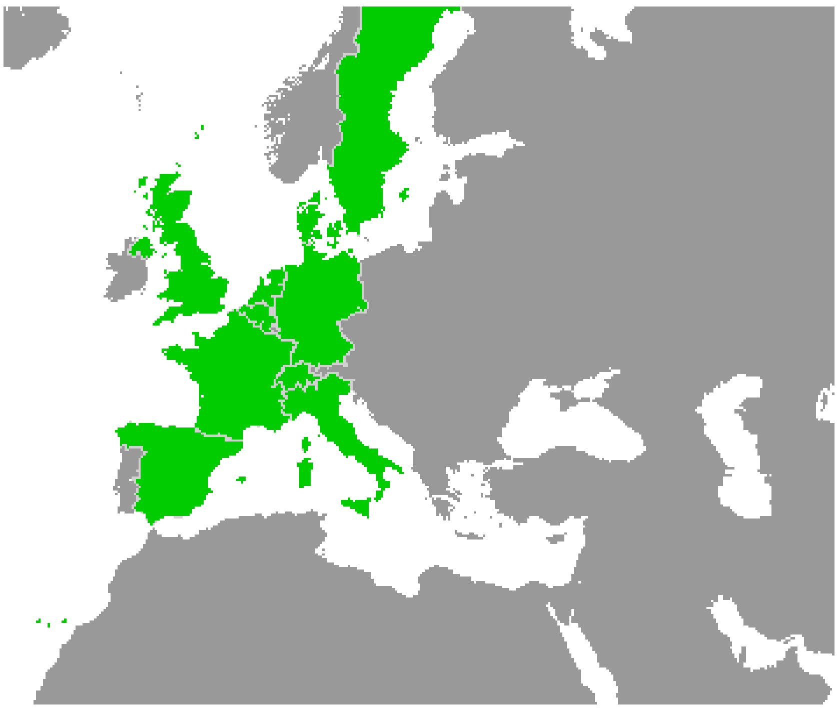
Work-Package 7: “Primary tool chain”

Evaluation of the models and tools against the WP2 requirements

List of criteria on means, models and tools and results on the benchmark

Marielle Petit-Doche

May 2013



This page is intentionally left blank

Work-Package 7: “Primary tool chain”

OETCS/WP7/O7.1.3_O7.1.7 – 00/02
May 2013

Evaluation of the models and tools against the WP2 requirements

List of criteria on means, models and tools and results on the benchmark

Marielle Petit-Doche

Systerel

Prepared for ITEA2 openETCS consortium
Europa

Abstract: This document gives elements to evaluate the means of modeling and the associated tools according WP2 requirements. Evaluation on the models and tools of benchmark is also described.

Table of Contents

Figures and Tables

Document information	
Work Package	WP7
Deliverable ID or doc. ref.	O7.3.1_O7.1.7
Document title	Evaluation of the models and toolsagainst the WP2 requirements
Document version	00.02
Document authors (org.)	Marielle Petit-Doche (Systerel)

Review information	
Last version reviewed	00.01
Main reviewers	Uwe Steinke (Siemens) Armand Nachef (CEA) Cyril Cornu (All4Tech) Alexandre Ginisty (All4Tech) Mathieu Perrin (CEA)

Approbation			
	Name	Role	Date
Written by	Marielle Petit-Doche	WP7-T7.1 Sub-Task Leader	
Approved by	Michael Jastram	WP7 leader	

Document evolution			
Version	Date	Author(s)	Justification
00.01	19/04/2013	M. Petit-Doche	Document creation by merging O7.1.3 and O7.1.7
00.02	02/05/2013	M. Petit-Doche	Review remarks Tool evaluation matrix

1 Introduction

The aim of this document is to report the results of the evaluation of the means of description to model the requirements of SUBSET-026 concerning the on-board unit and their associated tools.

This evaluation task is part of work package WP7, task 1 "Primary tool Chain analyses and recommendations". According to the results of WP2, especially the OpenETCS process and the requirements on language and tools, the aim of this task is to determine the best candidates to produce models of the on-board units, following the OpenETCS process

This process is described in details in D2.3 " Description of the openETCS process" and is summed up in the figure 1. Requirements references quoted in the current document are defined in D2.6 "Requirements for openETCS".

Yellow elements are inputs, blue elements are part of the design process, red elements are verification and validation activities, green elements are safety activities. Each line (between dash or full blue lines) is a phase of the process, with a name on the right.

The chapter 2 of this document provides a template to describe the means and tools and a list of criteria according WP2 requirements on language, models and tools. The objectives of this description and criteria are to allow to determine the best means of description and associated tool for a given activities.

The chapter 3 resumes the results of the evaluation at the end of the benchmark activities.

In Appendix, a chapter is dedicated to each models produced during the benchmark activities :

- GOPRR
- CORE
- ERTMSFormalSpecs
- SysML with Papyrus
- SysML with Entreprise Architect
- SCADE
- EventB with Rodin
- Classical B with Atelier B
- Petri Nets
- System C
- Why3
- GNATprove

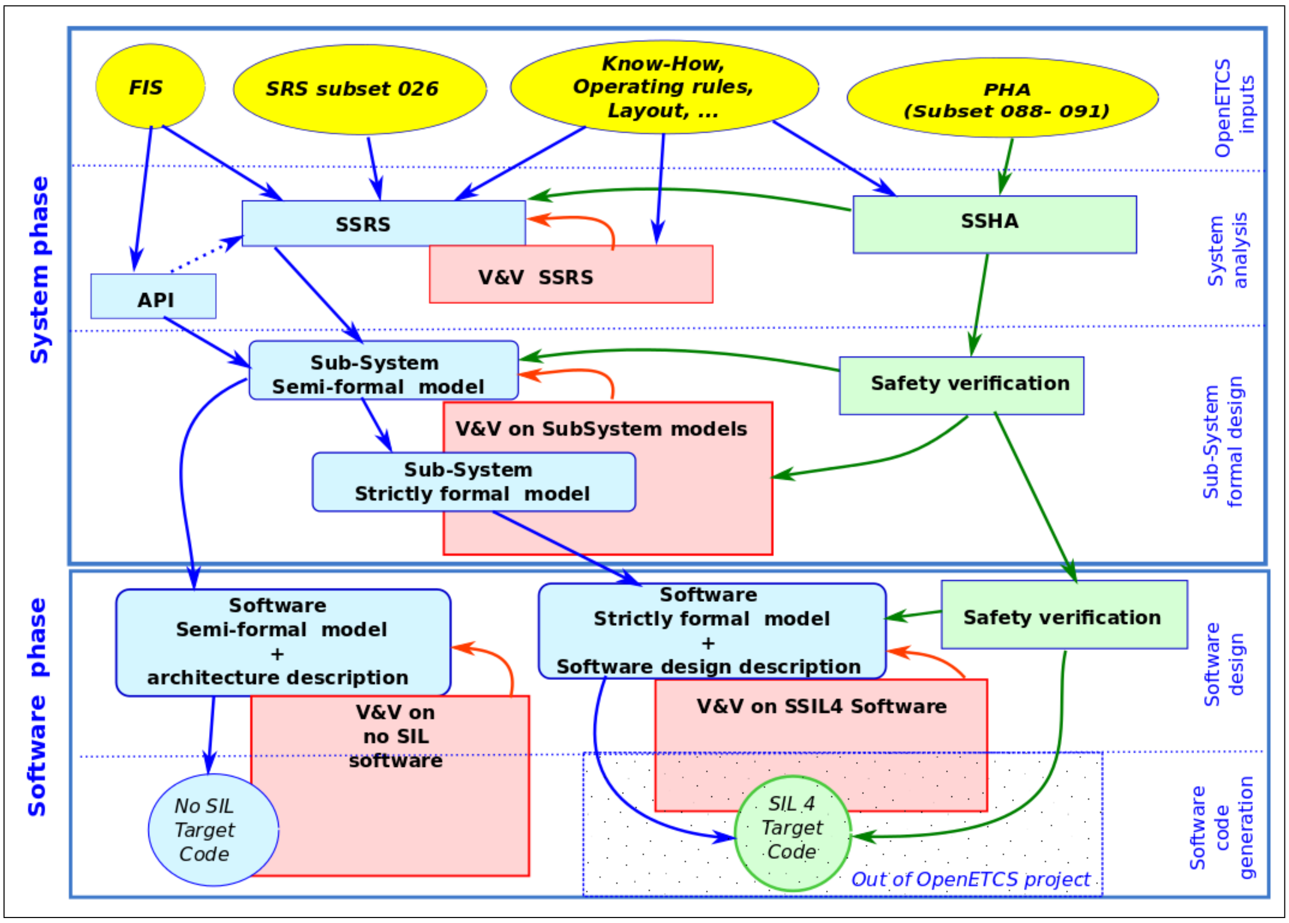


Figure 1. Main OpenETCS process

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.

Tool platform are not covered by this document but in an other output of WP7 : O7.1.9 "Evaluation of each tool platform against WP2 requirements, independent of target tools". Besides, Task 7.1 is focussing on design activities : despite that some means can provide verification artefacts for example, tools and means for validation, verification, test generation,... are in the scope of task 2 and will be analysed later.

1.1 Reference Documents

- CENELEC EN 50126-1 — 01/2000 — *Railways applications — The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) — Part 1: Basic requirements and generic process*
- CENELEC EN 50128 — 10/2011 — *Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems*
- CENELEC EN 50129 — 05/2003 — *Railway applications — Communication, signalling and processing systems — Safety related electronic systems for signalling*
- D2.1 – Report on existing methodologies
- D2.2 – Report on CENELEC standards

- D2.3 – Definition of the overall process for the formal description of ETCS and the rail system it works in
- D2.4 – Definition of the methods used to perform the formal description
- D2.6 – Requirements for OpenETCS

1.2 Glossary

API Application Programming Interface

FME(C)A Failure Mode Effect (and Criticity) Analysis

FIS Functional Interface Specification

HW Hardware

I/O Input/Output

OBU On-Board Unit

PHA Preliminary Hazard Analysis

QA Quality Analysis

RBC Radio Block Center

RTM RunTime Model

SIL Safety Integrity Level

SRS System Requirement Specification

SSHA Sub-System Hazard Analysis

SSRS Sub-System Requirement Specification

SW Software

THR Tolerable Hazard Rate

V&V Verification & Validation

2 Templates

Author Author of the approaches description %%Name - Company%%

Assessor 1 First assessor of the approaches %%Name - Company%%

Assessor 2 Second assessor of the approaches %%Name - Company%%

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

2.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name %%Name of the approach and the tool%%

Web site %%if available, how to find information%%

Licence %%Kind of licence%%

Abstract

Short abstract on the approach and tool (10 lines max)

Publications

Short list of publications on the approach (5 max)

2.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis				
Sub-system formal design				
Software design				
Software code generation				

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation				
Modeling				
Design				
Code generation				
Verification				
Validation				
Safety analyses				

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

2.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language				
Semi-formal language				
Formal language				
Structured language				
Modular language				
Textual language				
Mathematical symbols or code				
Graphical language				

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)				
Simple formalization of properties (D.2.6-X-28.1)				
Scalability : capability to design large model				
Easily translatable to other languages (D.2.6-X-30)				
Executable directly (D.2.6-X-33)				
Executable after translation to a code (D.2.6-X-33) (precise if the translation is automatic)				
Simulation, animation (D.2.6-X-33)				
Easily understandable (D.2.6-X-27)				
Expertise level needed (0 High level, 3 few level)				
Standardization (D.2.6-X-29)				
Documented (D.2.6-X-29)				
Extensible language (D.2.6-01-28)				

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

Language usage

Describe the possible restriction on the language

2.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)				
System architecture design (D.2.6-X-10.2)				
System data flow identification (D.2.6-X-10.2.3)				
Sub-system focus (D.2.6-X-10.2.4)				
System interfaces definition (D.2.6-X-10.2.5)				
System requirement allocation (D.2.6-X-10.3)				
Traceability with SRS (D.2.6-X-10.5)				
Traceability with Safety activities (D.2.6-X-11)				

2.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

2.5.1 Semi-formal model

Concerning semi-formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)				
Coverage of SSRS (D.2.6-X-12.2.1)				
Coverage of SSHA (D.2.6-X-12.2.2)				
Management of requirement justification (D.2.6-X-12.2.3)				
Traceability to SSRS (D.2.6-X-12.2.5)				
Traceability of exported requirements (D.2.6-X-12.2.6)				
Simulation or animation (D.2.6-X-13 partial)				
Execution (D.2.6-X-13 partial)				
Extensible to strictly formal model (D.2.6-X-14.3)				
Easy to refine towards strictly formal model (D.2.6-X-14.4)				
Extensible and modular design (D.2.6-X-15)				
Extensible to software architecture and design (D.2.6-X-30)				

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)				
Safety properties formalisation (D.2.6-X-22)				
Logical expression (D.2.6-X-28.2.2)				
Timing constraints (D.2.6-X-28.2.3)				
Safety properties validation (D.2.6-X-23.2)				
Logical properties assertion (D.2.6-X-34)				
Check of assertions (D.2.6-X-34.1)				

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines				
Time-outs				
Truth tables				
Arithmetic				
Braking curves				
Logical statements				
Message and fields				

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

2.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)				
Coverage of SSRS (D.2.6-X-14.2)				
Traceability to SSRS (D.2.6-X-14.3)				
Extensible to software design (D.2.6-X-16)				
Safety function isolation (D.2.6-X-17)				
Safety properties formalisation (D.2.6-X-22)				
Logical expression (D.2.6-X-28.2.2)				
Timing constraints (D.2.6-X-28.2.3)				
Safety properties validation (D.2.6-X-23.3)				
Logical properties assertion (D.2.6-X-34)				
Proof of assertions (D.2.6-X-34.2)				

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines				
Time-outs				
Truth tables				
Arithmetic				
Braking curves				
Logical statements				
Message and fields				

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

2.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

2.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model				
Software architecture description				
Software constraints				
Traceability				
Executable				

2.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model				
Software architecture description				
Software constraints				
Traceability				
Executable				
Conformance to EN50128 § 7.2				
Conformance to EN50128 § 7.3				
Conformance to EN50128 § 7.4				

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming				
Fault detection & diagnostic				
Error detecting code				
Failure assertion programming				
Diverse programming				
Memorising executed cases				
Software error effect analysis				
Fully defined interface				
Modelling				
Structured methodology				

2.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods				
Modeling				
Modular approach (mandatory)				
Components				
Design and coding standards (mandatory)				
Strongly typed programming language				

2.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support				
Automatic translation				
Code Generation				
Model verification				
Test generation				
Simulation, execution, debugging				
Formal proof				

Modelling support

Does the tool provide a textual or a graphical editor ?

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

Model verification

Which verification on models are provided by the tool?

Test generation

Does the tool allow to generate tests ? For which purpose ?

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ?

Formal proof

Does the tool allow formal proof ? How ?

2.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)				
Portability to operating systems (D2.6-X-37)				
Cooperation of tools (D2.6-X-38)				
Robustness (D2.6-X-41)				
Modularity (D2.6-X-41.1)				
Documentation management (D.2.6-X-41.2)				
Distributed software development (D.2.6-X-41.3)				
Simultaneous multi-users (D.2.6-X-41.4)				
Issue tracking (D.2.6-X-41.5)				
Differences between models (D.2.6-X-41.6)				
Version management (D.2.6-X-41.7)				
Concurrent version development (D.2.6-X-41.8)				
Model-based version control (D.2.6-X-41.9)				
Role traceability (D.2.6-X-41.10)				
Safety version traceability (D.2.6-X-41.11)				
Model traceability (D.2.6-01-035)				
Tool chain integration				
Scalability				

2.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)				
Proof of correctness (D.2.6-01-42.03)				
Existing industrial usage				
Model verification				
Test generation				
Simulation, execution, debugging				
Formal proof				

Other elements for tool certification

2.11 Other comments

Please to give free comments on the approach.

3 Conclusion

This conclusion give a sum up of the evaluation results for each approach. The detailed results of each approach are given in the appendix.

The evaluation of Core (see Appendix B) and Why3 (see Appendix K) have been stopped by the authors and do not appear in this conclusion.

Minus mark "-" means this criteria as not been evaluated for this approach.

Star mark "*" means this criteria has been difficult to evaluate for this approach.

3.1 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

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

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

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

3.2 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Informal language	0		9		0				0	
Semi-formal language	0		9		3				9	
Formal language	8		3		9				4	
Structured language	9		9		9				9	
Modular language	9		6		8				9	
Textual language	0		7		9				9	
Mathematical symbols or code	0		0		9				6	
Graphical language	9		9		9				0*	

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Declarative formalization of properties (D.2.6-X-28)	7		5(2)		6				3	
Simple formalization of properties (D.2.6-X-28.1)	6		6		6				3	
Scalability : capability to design large model	8		5*		8				9	
Easily translatable to other languages (D.2.6-X-30)	9		7		9				6	
Executable directly (D.2.6-X-33)	0		0		9				6*	
Executable after translation to a code (D.2.6-X-33)	9		3*		9				9	
(precise if the translation is automatic)	8				9				2*	
Simulation, animation (D.2.6-X-33)	0		4*		9				9	
Easily understandable (D.2.6-X-27)	6		4		9				7	
Expertise level needed (0 High level, 3 few level)	6		4		6				4	
Standardization (D.2.6-X-29)	*		9		6				9	
Documented (D.2.6-X-29)	7		6		8				9	
Extensible language (D.2.6-01-28)	8		9		5				9	

3.3 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Independent System functions definition (D.2.6-X-10.2.1)	6		4		9				-	
System architecture design (D.2.6-X-10.2)	9		9		3				-	
System data flow identification (D.2.6-X-10.2.3)	9		6		8				-	
Sub-system focus (D.2.6-X-10.2.4)	9		9		6				-	
System interfaces definition (D.2.6-X-10.2.5)	9		9		8				-	
System requirement allocation (D.2.6-X-10.3)	8		9		7				-	
Traceability with SRS (D.2.6-X-10.5)	6		6		9				-	
Traceability with Safety activities (D.2.6-X-11)	6		6		9				-	

3.4 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

3.4.1 Semi-formal model

Concerning semi-formal model, how the WP2 requirements are covered ?

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Consistency to SSRS (D.2.6-X-12.2)	-		6		-				9	
Coverage of SSRS (D.2.6-X-12.2.1)	-		5		-				2*	
Coverage of SSHA (D.2.6-X-12.2.2)	-		5		-				1*	
Management of requirement justification (D.2.6-X-12.2.3)	-		9		-				6	
Traceability to SSRS (D.2.6-X-12.2.5)	-		8		-				2*	
Traceability of exported requirements (D.2.6-X-12.2.6)	-		8		-				2*	
Simulation or animation (D.2.6-X-13 partial)	-		4*		-				9	
Execution (D.2.6-X-13 partial)	-		0*		-				9	
Extensible to strictly formal model (D.2.6-X-14.3)	-		3		-				4	
Easy to refine towards strictly formal model (D.2.6-X-14.4)	-		4		-				6	
Extensible and modular design (D.2.6-X-15)	-		9		-				9	
Extensible to software architecture and design (D.2.6-X-15)	-		8		-				9	

Concerning safety properties management, how the WP2 requirements are covered ?

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Safety function isolation (D.2.6-X-17)	-		9		-				*	
Safety properties formalisation (D.2.6-X-22)	-		4*		-				3*	
Logical expression (D.2.6-X-28.2.2)	-		5		-				9	
Timing constraints (D.2.6-X-28.2.3)	-		3		-				9	
Safety properties validation (D.2.6-X-23.2)	-		0		-				9	
Logical properties assertion (D.2.6-X-34)	-		3		-				9	
Check of assertions (D.2.6-X-34.1)	-		0		-				9	

Does the language allow to formalize (D.2.6-X-31):

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
State machines	-		8		-				8	
Time-outs	-		6		-				9	
Truth tables	-		3		-				9	
Arithmetic	-		0		-				9	
Braking curves	-		0		-				9	
Logical statements	-		7		-				9	
Message and fields	-		9		-				9	

3.4.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Consistency to SFM (D.2.6-X-14.2)	*		-		9				-	
Coverage of SSRS (D.2.6-X-14.2)	3		-		9				-	
Traceability to SSRS (D.2.6-X-14.3)	7		-		9				-	
Extensible to software design (D.2.6-X-16)	3		-		9				-	
Safety function isolation (D.2.6-X-17)	3		-		8				-	
Safety properties formalisation (D.2.6-X-22)	5		-		6				-	
Logical expression (D.2.6-X-28.2.2)	9		-		9				-	
Timing constraints (D.2.6-X-28.2.3)	0		-		8				-	
Safety properties validation (D.2.6-X-23.3)	7		-		6				-	
Logical properties assertion (D.2.6-X-34)	9		-		6				-	
Proof of assertions (D.2.6-X-34.2)	8		-		7				-	

Does the language allow to formalize (D.2.6-X-32):

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
State machines	9		-		9				-	
Time-outs	0		-		8				-	
Truth tables	0		-		9				-	
Arithmetic	9		-			9			-	
Braking curves	9		-		9				-	
Logical statements	9		-		9				-	
Message and fields	9		-		8				-	

3.5 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

3.5.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Derivation from system semi-formal model	*		9		9				4*	
Software architecture description	9		8		9				9	
Software constraints	9		6		8				9	
Traceability	9		6		9				6	
Executable	8		2*		9				9	

3.5.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Derivation from system semi-formal or strictly formal model	*		3		9				2*	
Software architecture description	9		9		9				9	
Software constraints	9		4		9				9	
Traceability	9		5		9				6	
Executable	8		0*		9				9	
Conformance to EN50128 § 7.2	0		5*		9				0*	
Conformance to EN50128 § 7.3	0		4*		9				3*	
Conformance to EN50128 § 7.4	0		4*		9				3*	

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Defensive programming	7		0		6				*	
Fault detection & diagnostic	7		0		0				6	
Error detecting code	7		0		0				9	
Failure assertion programming	9		6		1				9	
Diverse programming	3		0		0				*	
Memorising executed cases	0		0		3				9	
Software error effect analysis	0		2		0				2*	
Fully defined interface	9		9		9				9	
Modelling	9		8		9				5*	
Structured methodology	9		8		9				3*	

3.6 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Formal methods	6*		0		8				0	
Modeling	6*		4		9				9	
Modular approach (mandatory)	9		6		9				9	
Components	9		3		9				9	
Design and coding standards (mandatory)	6*		1		8				6*	
Strongly typed programming language	9		9		9				6	

3.7 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Modelling support	9		9		9				9	
Automatic translation	3		2		9				9	
Code Generation	9		0		9				2*	
Model verification	6		0		7				4*	
Test generation	*		5		4				2	
Simulation, execution, debugging	6		0		9				9	
Formal proof	0		0		7				5	

3.8 Use of the tool

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Open Source (D2.6-X-36)	0		9		0				9	
Portability to operating systems (D2.6-X-37)	9		6		4				9	
Cooperation of tools (D2.6-X-38)	3		8		8				4*	
Robustness (D2.6-X-41)	6*		3		9				9	
Modularity (D2.6-X-41.1)	0		8		9				9	
Documentation management (D.2.6-X-41.2)	0		5		9				4*	
Distributed software development (D.2.6-X-41.3)	9		6		6				9	
Simultaneous multi-users (D.2.6-X-41.4)	9		6		3				9	
Issue tracking (D.2.6-X-41.5)	0		4		0				3*	
Differences between models (D.2.6-X-41.6)	0		4		9				8	
Version management (D.2.6-X-41.7)	3		5		9				7	
Concurrent version development (D.2.6-X-41.8)	0		6		4*				7	
Model-based version control (D.2.6-X-41.9)	0		3		9				0*	
Role traceability (D.2.6-X-41.10)	0		4		0				0*	
Safety version traceability (D.2.6-X-41.11)	0		5		0				0*	
Model traceability (D.2.6-01-035)	3		7		9				0*	
Tool chain integration	6*		9		6*				8	
Scalability	6*		6		9				9	

3.9 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Tool manual (D.2.6-01-42.02)	9		7		9				6*	
Proof of correctness (D.2.6-01-42.03)	0		1		9				0*	
Existing industrial usage	9		5		9				9	
Model verification	3		0		8				0*	
Test generation	0		5		6				0*	
Simulation, execution, debugging	0		5		9				9	
Formal proof	0		0		7				0*	

Appendix A: GOPRR

Author Author of the approaches description Johannes Feuser/Cécile Braunstein (Uni. Bremen)

Assessor 1 First assessor of the approaches Alexandre Ginisty (All4Tec)

Assessor 2 Second assessor of the approaches Matthias Güdemann (Systerel)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

A.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name An openETCS donain specific language Based on GOPPRR

Web site http://www.informatik.uni-bremen.de/agbs/jfeuser/

Licence GPL V3

Abstract

The approach proposes an open domain specific language for openETCS. The language is based on the meta-meta-model GOPPRR, and the tool used is MetaEdit+.

Publications

Short list of publications on the approach (5 max)

- "MetaEdit+ Workbench User's Guide" accessed April 27, 2011. [Online].: http://www.metacase.com/support/45/m
- S. Kelly and J.-P. Tolvanen, "Domain-Specific Modeling". JOHN WILEY & SONS, INC., 2008.
- J. Feuser and J. Peleska, "Model Based Development and Tests for openETCS Applications – A Comprehensive Tool Chain", 12 2012, in in Proceedings of FORMS/FORMAT 2012.
- J. Feuser, "Open source Software for Train Control and Application and its Architectural Implication", PhD. Thesis, 2013.

A.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	2	2	1	5
Sub-system formal design	3	3	3	9
Software design	3	3	3	9
Software code generation	3	3	3	9

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	1	1	1	3
Modeling	3	3	3	9
Design	2	2	2	6
Code generation	3	3	3	9
Verification	0	0	0	0
Validation	0	0	0	0
Safety analysis	0	0	0	0

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

A.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0	0	0	0
Semi-formal language	0	0	0	0
Formal language	3	2	3	8
Structured language	3	3	3	9
Modular language	3	3	3	9
Textual language	0	0	0	0
Mathematical symbols or code	0	0	0	0
Graphical language	3	3	3	9

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	3	2	2	7
Simple formalization of properties (D.2.6-X-28.1)	2	2	2	6
Scalability : capability to design large model	3	3	2	8
Easily translatable to other languages (D.2.6-X-30)	3	3	3	9
Executable directly (D.2.6-X-33)	0	0	0	0
Executable after translation to a code (D.2.6-X-33)	3	3	3	9
(precise if the translation is automatic)	3	2	3	8
Simulation, animation (D.2.6-X-33)	0	0	0	0
Easily understandable (D.2.6-X-27)	2	2	2	6
Expertise level needed (0 High level, 3 few level)	2	2	2	6
Standardization (D.2.6-X-29)	*	*	*	*
Documented (D.2.6-X-29)	3	2	2	7
Extensible language (D.2.6-01-28)	3	3	2*	8

Assessor 2 wrt. extensibility: the language is designed to be specific for ETCS modeling.

Author: (*) The meta-meta model (GOPPRR) is not a standard but it is formally defined.

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

The language is fully documented, and the meta-model of the language is given.

Language usage

Describe the possible restriction on the language

A.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)	2	2	2	6
System architecture design (D.2.6-X-10.2)	3	3	3	9
System data flow identification (D.2.6-X-10.2.3)	3	3	3	9
Sub-system focus (D.2.6-X-10.2.4)	3	3	3	9
System interfaces definition (D.2.6-X-10.2.5)	3	3	3	9
System requirement allocation (D.2.6-X-10.3)	3	3	2	8
Traceability with SRS (D.2.6-X-10.5)	2	2	2	6
Traceability with Safety activities (D.2.6-X-11)	2	2	2	6

A.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

A.5.1 Semi-formal model

Comment. Section has been skipped.

A.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	*	*	*	*
Coverage of SSRS (D.2.6-X-14.2)	1	1	1	3
Traceability to SSRS (D.2.6-X-14.3)	3	2	2	7
Extensible to software design (D.2.6-X-16)	1	1	1	3
Safety function isolation (D.2.6-X-17)	1	1	1	3
Safety properties formalisation (D.2.6-X-22)	2	2	1	5
Logical expression (D.2.6-X-28.2.2)	3	3	3	9
Timing constraints (D.2.6-X-28.2.3)	0	0	0	0
Safety properties validation (D.2.6-X-23.3)	3	2	2	7
Logical properties assertion (D.2.6-X-34)	3	3	3	9
Proof of assertions (D.2.6-X-34.2)	3	3	2	8

Author: *(*) Since the semi-formal model is not yet done, it is hard to say.*

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3	3	9
Time-outs	0	0	0	0
Truth tables	0	0	0	0
Arithmetic	3	3	3	9
Braking curves	3	3	3	9
Logical statements	3	3	3	9
Message and fields	3	3	3	9

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

A.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

A.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	*	*	*	*
Software architecture description	3	3	3	9
Software constraints	3	3	3	9
Traceability	3	3	3	9
Executable	3	2	3	8

Author: (*) Since we do not know the format of the semi-formal model yet, it is hard to say.

A.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	*	*	*	*
Software architecture description	3	3	3	9
Software constraints	3	3	3	9
Traceability	3	3	3	9
Executable	3	2	3	8
Conformance to EN50128 § 7.2	0	0	0	0
Conformance to EN50128 § 7.3	0	0	0	0
Conformance to EN50128 § 7.4	0	0	0	0

> *Author: (*) Since we do not know the format of the semi-formal model yet, it is hard to say.*

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming	3	3	1*	7*
Fault detection & diagnostic	3	3	1	7
Error detecting code	3	3	1	7
Failure assertion programming	3	3	3	9
Diverse programming	1	1	1	3
Memorising executed cases	0	0	0	0
Software error effect analysis	0	0	0	0
Fully defined interface	3	3	3	9
Modeling	3	3	3	9
Structured methodology	3	3	3	9

Assessor 2 I am not aware of explicit means to defensive programming, nor are they mentioned in Johannes' thesis.

A.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	3	3	*	6*
Modeling	3	3	*	6*
Modular approach (mandatory)	3	3	3	9
Components	3	3	3	9
Design and coding standards (mandatory)	3	3	*	6*
Strongly typed programming language	3	3	3	9

Assessor 2 Are there coding standards?

A.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3	3	3	9
Automatic translation	1	1	1	3
Code Generation	3	3	3	9
Model verification	2	2	2	6
Test generation	*	*	*	*
Simulation, execution, debugging	2	2	2	6
Formal proof	0	0	0	0

Author: (*) The new version of MetaEdit+ provides this feature.

Modeling support

Does the tool provide a textual or a graphical editor ?

Graphical

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

MERL generator

Model verification

Which verification on models are provided by the tool?

Inspection

Test generation

Does the tool allow to generate tests ? For which purpose ?

No

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ?

The new version yes.

Formal proof

Does the tool allow formal proof ? How ?

No

A.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	0	0	0	0
Portability to operating systems (D2.6-X-37)	3	3	3	9
Cooperation of tools (D2.6-X-38)	1	1	1	3
Robustness (D2.6-X-41)	3	3	*	6*
Modularity (D2.6-X-41.1)	0	0	0	0
Documentation management (D.2.6-X-41.2)	0	0	0	0
Distributed software development (D.2.6-X-41.3)	3	3	3	9
Simultaneous multi-users (D.2.6-X-41.4)	3	3	3	9
Issue tracking (D.2.6-X-41.5)	0	0	0	0
Differences between models (D.2.6-X-41.6)	0	0	0	0
Version management (D.2.6-X-41.7)	1	1	1	3
Concurrent version development (D.2.6-X-41.8)	0	0	0	0
Model-based version control (D.2.6-X-41.9)	0	0	0	0
Role traceability (D.2.6-X-41.10)	0	0	0	0
Safety version traceability (D.2.6-X-41.11)	0	0	0	0
Model traceability (D.2.6-01-035)	1	1	1	3
Tool chain integration	3	3	*	6*
Scalability	3	3	*	6*

Assessor 2 Could not test the robustness of Metaedit+.

Seems to be a stand-alone tool, so toolchain integration is difficult to assess without trying the tool.

A.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3	3	3	9
Proof of correctness (D.2.6-01-42.03)	0	0	0	0
Existing industrial usage	3	3	3	9
Model verification	1	1	1	3
Test generation	0	0	0	0
Simulation, execution, debugging	0	0	0	0
Formal proof	0	0	0	0

Other elements for tool certification

A.11 Other comments

Please to give free comments on the approach.

Appendix B: CORE

Author Author of the approaches description: Cyril Cornu (All4tec)

B.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name: Core Workstation 5.1

Web site: `http://www.vitechcorp.com/products/core.shtml`

Licence: Shareware

Abstract

Short abstract on the approach and tool (10 lines max)

CORE is a comprehensive modeling environment built for complex systems engineering problems and based on EFFBD structured language. It integrates graphical modeling capabilities to assess and control design and program risks. By linking all elements of a system through a central model, a greater visibility is then provided into drivers for risk and system weaknesses. It allows building better models and delivering better products to market through:

- Integrated requirements management
- Fully executable behavior models
- Architecture development tools
- Validation and Verification (simulation)
- Comprehensive system documentation

Publications

Short lisvenduabandonnét of literature on the approach (5 max)

- Model-Based Systems Engineering by Vitech `http://www.mbseprimer.com/`
- System Engineering & Architecting with CORE `http://fr.scribd.com/doc/49887858/07-02-28-Vitech-CORE`

B.2 Evaluation

The evaluation of this approach has been stop by the author before the end of the benchmark activity.

Author:

The Main reasons to stop CORE tool evaluation are the following:

- *The Software is not Open Source, and can not be used for free in industrial fields of for industrial purposes. Moreover, this tool is no more provided in France, where just old versions are distributed and sold.*
- *The Software has very few possibilities for interfacing with other tools, thanks to its proprietary EFFBD language. Therefore, interface this tool with specific model checker or secondary Toolchain software (for instance Model Based Testing solution, of Safety Analysis tool) would need specific gateway development, that All4tec could not provide in the framework of All4tec contribution within OpenETCS. Therefore, a major part of CORE models added avalue would be lost.*
- *This software is supposed to model scenarii based on user approach of the system. Such an approach can be started only once the operating rules are provided as input. These inputs are still incomplete for the project Open ETCS, and the Open ETCS process is willing to base its model on Sub-System Requirements Specification, currently under construction.*
- *For all previous reasons, All4tec has decided to focus its effort on modeling and toolchain development on the Papyrus SysML/ UML Modeler tool. Indeed, the partnership with the CEA and the Papyrus tool developers, and the amount of perspectives for interfacing Papyrus with other part of the toolchain are supporting this decision as well.*

All4tec keeps although the possibility to support its system Safety analysis or functional approaches with this tool, but no Core development perspectives have to be expected for the Open ETCS project.

Appendix C: ERTMSFormalSpecs

Author Author of the approaches description Stanislas Pinte (ERTMS Solutions)

Assessor 1 First assessor of the approaches Renaud De Landtsheer (Alstom Be)

Assessor 2 Second assessor of the approaches Marielle Petit-Doche (Systerel)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted, strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

C.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name ERTMSFormalSpecs

Web site https://www.ertmssolutions.com/ertms-formalspecs/

License EUPL (https://github.com/openETCS/ERTMSFormalSpecs)

Abstract

Short abstract on the approach and tool (10 lines max)

ERTMSFormalSpecs provides a domain-specific language, designed to express the ERTMS specification in a concise and verifiable formal representation. It is understandable by domain specialists while retaining the ability to be translated to executable representations by fully automated means.

Assessor 1: From my experience, a domain-specific language really provides an interesting productivity gain, provided two conditions are met: first: the language should indeed be adapted to the domain, which seems to be the case given the covertness of the subset26 by the models written in this formalism, second: that something can be done downwards with this model, such as code generation, or translation to some other model, or a better understanding of the original document.

Publications

Short list of publications on the approach (5 max)

`http://www.ertmssolutions.com/files/ERTMSFormalSpecs_WCRR2011.pdf` `http://www.ertmssolutions.com/files/UsingERTMSFormalSpecsToModelBrakingCurves.pdf`

C.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	0	1		
Sub-system formal design	3	3		
Software design	0	0		
Software code generation	0	0		

Author: The scope of ERTMSFormalSpecs is, as described above, "a domain-specific language, designed to express the ERTMS specification in a concise and verifiable formal representation." Therefore, it is targeted to be used for sub-system formal design, not for analysis, software design or software code generation.

Assessor 1: Software code could be generated from such models, eg for simulation purposes, given that the semantics is executable. This could be done from the exports provided by ERTMSFormalSpec (XML or EMF/Eclipse).

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	3	3		
Modeling	3	3		
Design	3	3		
Code generation	0	1		
Verification	3	3		
Validation	3	3		
Safety analyses	0	0		

Assessor 1: ERTMSFormalSpec incorporates a simulator, which can be driven step by feeding it with test cases. The ERTMS tooling performs a limited set of validation described in section 6.9 "'Model validations"' of the document EFSW_User_Guide.pdf. These are rather syntactic validation, and not behavioral ones. More validation can be done through testing. Test cases incorporate expectations that are checked during the execution of the test case. Notice that there is no mechanism to attach expectation to the model, e.g. to a state, or state machine. ERTMSFormalSpec provides a covertness report mechanism for test cases, to identify which rule has been executed during a test suite.

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

Author: The ERTMSFormalSpecs approach has been designed specifically for the purpose of modelling an OBU application software. The ERTMSFormalSpecs approach is 100% aligned with the OpenETCS objective 1, which is to have a 100% semi-formal model of the SSRS.

C.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0	0		
Semi-formal language	3	3		
Formal language	3	3		
Structured language	3	3		
Modular language	3	3		
Textual language	3	2		
Mathematical symbols or code	3	3		
Graphical language	3	3		

Assessor 1: Regarding the "'Graphical Language"' item, ERTMSFormalSpec provides a graphical rendering of the state machines present in the model, and these representation

are editable. Besides, I did not see a graphical rendering of the data flows between components i.e.: an architectural view.

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	0	1		
Simple formalization of properties (D.2.6-X-28.1)	2	0		
Scalability : capability to design large model	3	3		
Easily translatable to other languages (D.2.6-X-30)	3	3		
Executable directly (D.2.6-X-33)	3	3		
Executable after translation to a code (D.2.6-X-33)	3	3		
(precise if the translation is automatic)	2	1		
Simulation, animation (D.2.6-X-33)	3	3		
Easily understandable (D.2.6-X-27)	3	2		
Expertise level needed (0 High level, 3 few level)	2	2		
Standardization (D.2.6-X-29)	3	2		
Documented (D.2.6-X-29)	3	3		
Extensible language (D.2.6-01-28)	3	2		

Assessor 1: ERTMSFormalSpec models are easily translatable to other language, but I fear that the language is so rich and domain-specific (braking curve feature) that the target language will need to be very rich as well to produce human-understandable models. Regarding the animation, I feel that a more structured insight could be given about the model during the simulation, e.g. illustrating the interaction between modules of the model. The extensibility of the language seems to arise from the openness of the supporting tool. No extension mechanism is described in the documentation (EFSW_Technical_Design.pdf, and EFSW_User_Guide.pdf)

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

ERTMSFormalSpecs provides the following documentation set:

- EFSW_Release_Notes.pdf (`https://github.com/openETCS/ERTMSFormalSpecs/blob/master/ErtmsFormalSpecs/doc/EFSW_Release_Notes.pdf`)
- EFSW_Technical_Design.pdf (`https://github.com/openETCS/ERTMSFormalSpecs/blob/master/ErtmsFormalSpecs/doc/EFSW_Technical_Design.pdf`)
- EFSW_User_Guide.pdf (`https://github.com/openETCS/ERTMSFormalSpecs/blob/master/ErtmsFormalSpecs/doc/EFSW_User_Guide.pdf`)
- ERTMSFormalSpecs-Tutorial (`https://github.com/openETCS/ERTMSFormalSpecs/wiki/ERTMSFormalSpecs-Tutorial`)

- ERTMSFormalSpecs-FAQ (`https://github.com/openETCS/ERTMSFormalSpecs/wiki/ERTMSFormalSpecs-FAQ`)

Language usage

Describe the possible restriction on the language

C.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)	3	3		
System architecture design (D.2.6-X-10.2.2)	3	3		
System data flow identification (D.2.6-X-10.2.3)	3	2		
Sub-system focus (D.2.6-X-10.2.4)	3	3		
System interfaces definition (D.2.6-X-10.2.5)	3	3		
System requirement allocation (D.2.6-X-10.3)	3	3		
Traceability with SRS (D.2.6-X-10.5)	3	3		
Traceability with Safety activities (D.2.6-X-11)	0	0		

Author: Although ERTMSFormalSpecs is not made for system analysis, it can be used for the following aspects on system level: Modelling of (separate) system functions, data flows, state machines and interfaces. It also provides complete SRS traceability support.

Assessor 1: Although the data flows between components, and the interfaces of components (which are a fragment of the data flows) are represented in the model, I do not see a simple (e.g. graphical) rendering of them, so a user needing this information in a synthetic way might need to extract it by hand with the current version of the tool. I do believe that this feature is rather easy to add, so my score is 2 for (D.2.6-X-10.2.3).

C.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focus on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

C.5.1 Semi-formal model

Author: ERTMSFormalSpecs models are formal in the sense that the ERTMSFormalSpecs language is fully defined with a grammar and complete semantics. However, they are *semi-formal in the sense that there is no mathematical proof theory at the basis of the language definition.*

Concerning semi-formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	3	3		
Coverage of SSRS (D.2.6-X-12.2.1)	3	3		
Coverage of SSHA (D.2.6-X-12.2.2)	3	3		
Management of requirement justification (D.2.6-X-12.2.3)	3	3		
Traceability to SSRS (D.2.6-X-12.2.5)	3	3		
Traceability of exported requirements (D.2.6-X-12.2.6)	3	3		
Simulation or animation (D.2.6-X-13 partial)	3	3		
Execution (D.2.6-X-13 partial)	3	3		
Extensible to strictly formal model (D.2.6-X-14.3)	3	3		
Easy to refine towards strictly formal model (D.2.6-X-14.4)	3	2		
Extensible and modular design (D.2.6-X-15)	3	3		
Extensible to software architecture and design (D.2.6-X-30)	3	3		

Assessor 1: Concerning "'Extensible to strictly formal model (D.2.6-X-14.3)"', This perfectly illustrates the usefulness of a domain-specific language, where things are easy to represent, and where the formalization process forces one to resolve all form of unclear features of the requirements specification.

Concerning "'Easy to refine towards strictly formal model (D.2.6-X-14.4)"', the D2.6 mentions a transformation process. Since such process can be implemented as soon as a formal semantics is available. Since this process is not available as from today, I have to put a mark 2.

Concerning "'Extensible and modular design (D.2.6-X-15)"', the language is itself modular and extensible, however, this modularity and extensibility could be made more efficient if synthetic views on the structure of the model were available (see my remark on data-flows and interface above).

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	1	1		
Safety properties formalisation (D.2.6-X-22)	2	2		
Logical expression (D.2.6-X-28.2.2)	2	3		
Timing constraints (D.2.6-X-28.2.3)	2	2		
Safety properties validation (D.2.6-X-23.2)	2	2		
Logical properties assertion (D.2.6-X-34)	2	3		
Check of assertions (D.2.6-X-34.1)	2	2		

Author: ERTMSFormalSpecs is a modelling language for functions. Therefore, only the functional aspects of properties are addressed.

Assessor 1: ERTMSFormalSpec includes the possibility to declaratively state a property, called "‘expectation"’ attached to a test step. Their truth value is evaluated during the test execution. They can encompass logical conditions, deadlines, and some state-machine specific conditions. Timing constraints are restricted to deadline enforcement. More intricate conditions on time cannot be represented, as they would require the use of non-available temporal operators.

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3		
Time-outs	3	3		
Truth tables	3	2+		
Arithmetic	3	3		
Braking curves	3	3		
Logical statements	3	3		
Message and fields	3	3		

Assessor 1: Truth tables can de represented as functions ranging on booleans, and returning a boolean. There is no graphical support for them, but this can be added very easily to the tool, hence I’ve put a mark 2+ for this criterion.

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

Author: ERTMSFormalSpecs has been designed to model the Subset-026 and test the S026 model, without taking Safety aspects into account initially.

C.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

Author: Even though ERTMSFormalSpecs models are formal, ERTMSFormalSpecs doesn’t aim to be used a strictly formal model, for proving purposes, in the context of the OpenETCS project. Therefore that section is skipped from evaluation.

Assessor 1: I skip this part as well

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)				
Coverage of SSRS (D.2.6-X-14.2)				
Traceability to SSRS (D.2.6-X-14.3)				
Extensible to software design (D.2.6-X-16)				
Safety function isolation (D.2.6-X-17)				
Safety properties formalisation (D.2.6-X-22)				
Logical expression (D.2.6-X-28.2.2)				
Timing constraints (D.2.6-X-28.2.3)				
Safety properties validation (D.2.6-X-23.3)				
Logical properties assertion (D.2.6-X-34)				
Proof of assertions (D.2.6-X-34.2)				

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines				
Time-outs				
Truth tables				
Arithmetic				
Braking curves				
Logical statements				
Message and fields				

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

C.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

Author: *ERTMSFormalSpecs scope is limited to modelling in the large (modelling, test and documentation). Therefore, software design section is skipped.*

Assessor 1: *I skip this part as well*

C.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model				
Software architecture description				
Software constraints				
Traceability				
Executable				

C.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model				
Software architecture description				
Software constraints				
Traceability				
Executable				
Conformance to EN50128 Â§ 7.2				
Conformance to EN50128 Â§ 7.3				
Conformance to EN50128 Â§ 7.4				

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming				
Fault detection & diagnostic				
Error detecting code				
Failure assertion programming				
Diverse programming				
Memorising executed cases				
Software error effect analysis				
Fully defined interface				
Modelling				
Structured methodology				

C.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Author: ERTMSFormalSpecs scope is limited to modelling in the large (modelling, test and documentation). Therefore, software code generation section is skipped.

Assessor 1: I skip this part as well

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods				
Modeling				
Modular approach (mandatory)				
Components				
Design and coding standards (mandatory)				
Strongly typed programming language				

C.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3	3		
Automatic translation	1	1		
Code Generation	1	1		
Model verification	3	3		
Test generation	1	1		
Simulation, execution, debugging	3	3		
Formal proof	0	0		

Modelling support

Does the tool provide a textual or a graphical editor ?

Author: Both.

Assessor 1: The editor is primarily a structured text editor where every bit of the model can be edited as a little text. A non-editable textual rendering is provided for some fragments of code, a.k.a. procedures, and a graphical editor is specifically available for state machines.

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

Author: As of today, the ERTMSFormalSpecs model is available both as an XML file and as an EMF model. The EMF model can be used to develop model-to-model translator (for instance ERTMSFormalSpecs->SCADE) or code generators.

Model verification

Which verification on models are provided by the tool?

> *Author: The ERTMSFormalSpecs model can be tested, by writing ERTMSFormalSpecs test cases in ERTMSFormalSpecs language (based on steps, actions and expectations), executing and debugging these test cases, and generating a test report.*
>
> *ERTMSFormalSpecs test cases can also be executed in an automated fashion for nightly build non-regression testing purposes.*

Test generation

Does the tool allow to generate tests ? For which purpose ?

> *Author: No, ERTMSFormalSpecs doesn't allow to generate tests. Integration between ERTMSFormalSpecs and RT-Tester is under study, to allow for automatic ERTMSFormal-Specs model test cases generation.*

Simulation, execution, debugging

Does the tool allow to simulate or to debug step by step a model or a code ?

> *Author: Yes, all of this is described in the ERTMSFormalSpecs User Guide.*

Formal proof

Does the tool allow formal proof ? How ?

> *Author: No, ERTMSFormalSpecs doesn't allow formal proof.*

C.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	3	3		
Portability to operating systems (D2.6-X-37)	2	3		
Cooperation of tools (D2.6-X-38)	3	3		
Robustness (D2.6-X-41)	3	3		
Modularity (D2.6-X-41.1)	3	3		
Documentation management (D.2.6-X-41.2)	2	2		
Distributed software development (D.2.6-X-41.3)	2	2		
Simultaneous multi-users (D.2.6-X-41.4)	1	1		
Issue tracking (D.2.6-X-41.5)	3	3		
Differences between models (D.2.6-X-41.6)	1	2		
Version management (D.2.6-X-41.7)	2	3		
Concurrent version development (D.2.6-X-41.8)	2	2		
Model-based version control (D.2.6-X-41.9)	2	2		
Role traceability (D.2.6-X-41.10)	1	1		
Safety version traceability (D.2.6-X-41.11)	0	0		
Model traceability (D.2.6-01-035)	3	3		
Tool chain integration	2	2		
Scalability	3	3		

C.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

Author: ERTMSFormalSpecs has EN50128 certifiability compliance outside of its scope, for the version available as of today. Certifiability compliance may be in the scope of future versions. However, relevant sections of this section are filled.

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3	3		
Proof of correctness (D.2.6-01-42.03)	2	1		
Existing industrial usage	0	0		
Model verification	3	3		
Test generation	0	0		
Simulation, execution, debugging	3	3		
Formal proof	0	0		

Other elements for tool certification

C.11 Other comments

Please to give free comments on the approach.

Assessor 1: To me, the main strengths of this approach are:

- *its very strong support for traceability to the Subset26, and to the Subset-076 for test cases*
- *its domain-specific language, which is*
  - *productive for this field, thanks to its expressivity, illustrated by the primitives developed for braking curves;*
  - *scalable, as is demonstrated by the large fraction of the Subset26 which has been represented so far.*

To me, the main weaknesses of this approach are:

- *its rather unfriendly look and feel, notably for the graphical editing of state machine, and for the global overview of the model, whose structure seems to match the one of the Subset26, and its lack of graphical rendering of the architecture, which would possibly ease the navigation in the model.*
- *its rather unfamiliar notation.*

The first can be improved rather easily, and the second can be coped with given that it was developped specifically to capture the Subset26. They should not be considered as blocking issues.

The ERTMSFormalSpec tooling is nearly exclusively centered on the model elaboration. Some tool integration should be developed to exploit the model developed with ERTMSFormalSpec. This can be done based on the exports of ERTMSFormalSpec (XML or EMF/Eclipse).

ERTMSFormalSpec is specifically devoted to the modeling of the Subset26 as it is which makes it a very credible and adapted tooling for the elaboration of a semi-formal model of this system. It is especially credible given the large fraction of the Subset26 that is already modeled in this framework.

Finally, all languages are a trade-off between what we want to express, and what we are able to handle (i.e.: compile, analyze, verify, etc). As a domain-specific language ERTMSFormalSpec is clearly choosing the expressivity over the handling, yet it remains an interpretable (but Turing-complete) language.

Appendix D: SysML with Papyrus

Author Author of the approaches description : CEA Nano-Innov labs (Agnes Lanusse, Mathieu Perrin and Armand Nachef) and All4tec (Alexandre Ginisty and Cyril Cornu)

Assessor 1 Renaud De Landsheer (Alstom BE)

Assessor 2 Marielle Petit-Doche (Systerel)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format

Assessor 1: First assessor can add comments using this format

Assessor 2: Second assessor can add comments using this format

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

D.1 Presentation

This section gives a quick presentation of the approach.

Name Name of the approach

Web site `http://www.eclipse.org/papyrus/`

Licence Open source : EUPL

Abstract

Short abstract on the approach and tool (10 lines max)

Publications

Short list of publications on the approach (5 max)

D.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	2	3	2	7
Sub-system formal design	3	1	2	6
Software design	2	2	2	6
Software code generation	1	1	1	3

Assessor 1: Concerning the "‘Sub-system formal design"’, I consider that Papyrus is more a semi-formal too than a fully formal one, so I’ve put a mark "‘2"’ for this criterion. As a matter of fact, it is quoted "‘1"’ for use as a formal language below.

Assessor 2: I agree comment of assesor 1, SysML is a "semi-formal" approach and does not allow design of sub-system "fully formal". Moreover expression of properties is poor in strict SysMl language. However SysML is good to produce a structured model of the system and to help in its analyses, but it shall be completed to cover all the needs of system analysis.

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	1	3	2	6
Modeling	3	3	3	9
Design	2	2	2	6
Code generation	1	1	1	3
Verification	3	2	1	6
Validation	2	2	1	5
Safety analyses	3	*	1	4*

Assessor 1: I’ve put a rather higher mark on Documentation because the models supported by Papyrus can be used as documentation items.

Concerning the safety analyses, I do not see how Papyrus helps in driving such analyses.

Assessor 2: SysML can provide useful elements for documentation, however, a clear methodology shall be provide to understand and drive the models. For verification, validation and safety analysis, SysML alone is not sufficient for critical systems.

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

Assessor 2: Elements missing

D.3 Language

This section discusses the main element of the language.

According WP2 requirements, give a note for the characteristics of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Informal language	3	3	3	9
Semi-formal language	3	3	3	9
Formal language	1	1	1	3
Structured language	3	3	3	9
Modular language	2	2	2	6
Textual language	3	2	2	7
Mathematical symbols or code	0	0	0	0
Graphical language	3	3	3	9

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	3 (0)	1	1	5(2)
Simple formalization of properties (D.2.6-X-28.1)	3	2	1	6
Scalability : capability to design large model	3	*	2	5*
Easily translatable to other languages (D.2.6-X-30)	2	3	2	7
Executable directly (D.2.6-X-33)	0	0	0	0
Executable after translation to a code (D.2.6-X-33) (precise if the translation is automatic)	1	2	*	3*
Simulation, animation (D.2.6-X-33)	2	2	*	4*
Easily understandable (D.2.6-X-27)	1	2	1	4
Expertise level needed (0 High level, 3 few level)	1	2	1	4
Standardization (D.2.6-X-29)	3	3	3	9
Documented (D.2.6-X-29)	2	2	2	6
Extensible language (D.2.6-01-28)	3	3	3	9

Assessor 1: In my understanding, if a language supports the declarative formalization of properties there should be the possibility to write some mathematical symbols, or code, so I do not understand the notations provided by the authors in the points "‘Declarative formalization of properties"’, and "‘Mathematical symbols or code"’. I see that declarative

properties could be attached to all state, as pre or post-condition, or as "‘constraints"’ to any ActivityFigure. The language encompass some timing properties, and seemignly arbitrary expressions, but again, the language seems partially formal. Also, the editing actions necessary to add such constraint is rather complex.

Assessor 2: Expression of properties is difficult in sysML, and the OCL language is very poor. Besides there are no good tool to check the properties.

I have no see example of execution, simulation,...

Concerning scalability, translation, understanding or expertise level, SysML can not be used with a good methodology guide (standard are not sufficient) to precisely define how and with which elements we can draw models. If a model as been defined without following a set of predefined rules, it can not be easily translate in an another language, or only partly.

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

Assessor 2: Elements missing

Language usage

Describe the possible restriction on the language

Assessor 2: Elements missing

D.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.1.1)	1	1	2	4
System architecture design (D.2.6-X-10.1.2)	3	3	3	9
System data flow identification (D.2.6-X-10.1.3)	2	2	2	6
Sub-system focus (D.2.6-X-10.1.4)	3	3	3	9
System interfaces definition (D.2.6-X-10.1.5)	3	3	3	9
System requirement allocation (D.2.6-X-10.2)	3	3	3	9
Traceability with SRS (D.2.6-X-10.3)	3	2	1	6
Traceability with Safety activities (D.2.6-X-11)	3	2	1	6

Assessor 1: concerning the "'Traceability with SRS"', this requires that the SSRS is encoded in the formalism of Papyrus. I just wonder how the Subset26 can be encoded with its structure, into the requirements formalism of Papyrus.

Assessor 2: In the given examples it is very difficult to check the traceability with the SRS or SSRS (lots of comments are missing). Besides, how can we check the coverage of SRS requirement by the SysML model ?

D.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focus on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

D.5.1 Semi-formal model

Concerning semi-formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	2	2	2	6
Coverage of SSRS (D.2.6-X-12.2.1)	2	2	1	5
Coverage of SSHA (D.2.6-X-12.2.2)	2	2	1	5
Management of requirement justification (D.2.6-X-12.2.3)	3	3	3	9
Traceability to SSRS (D.2.6-X-12.2.5)	3	3	2	8
Traceability of exported requirements (D.2.6-X-12.2.6)	3	3	2	8
Simulation or animation (D.2.6-X-13 partial)	2	2	*	4*
Execution (D.2.6-X-13 partial)	0	0	*	0*
Extensible to strictly formal model (D.2.6-X-14.3)	1	1	1	3
Easy to refine towards strictly formal model (D.2.6-X-14.4)	1	1	2	4
Extensible and modular design (D.2.6-X-15)	3	3	3	9
Extensible to software architecture and design (D.2.6-X-30)	3	3	2	8

Assessor 2: Same comments as below on execution and simulation.

Current state of SysML language do not allow a good management of the traceability, expecially coverage topics.

It is difficult to extend SysML to strictly formal model, however it can be refined to a formal model on some parts if some methodological constraints have been followed.

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	3	3	3	9
Safety properties formalisation (D.2.6-X-22)	3	*	1	4*
Logical expression (D.2.6-X-28.2.2)	3	1	1	5
Timing constraints (D.2.6-X-28.2.3)	1	1	1	3
Safety properties validation (D.2.6-X-23.2)	0	0	0	0
Logical properties assertion (D.2.6-X-34)	3	0	0	3
Check of assertions (D.2.6-X-34.1)	0	0	0	0

Assessor 2: SysML alone is to poor to deal with safety aspects.

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	2	3	3	8
Time-outs	2	2	2	6
Truth tables	1	1	1	3
Arithmetic	0	0	0	0
Braking curves	0	0	0	0
Logical statements	3	3	1	7
Message and fields	3	3	3	9

Assessor 1: I feel a bit uncomfortable saying what can be formalized in a semi-formal way, so I'v interpreted this as "‘represented"’ in a semi-formal way.

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

Assessor 2: I think SysML is good to give a clear structure of the system and its architecture, but insufficient to cover all the process and safety aspects.

D.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

Assessor 1: SysML/Papyrus is obviously not a strictly formal tool, I therefor skip this section. This is illustrated by the evaluation of the author in the second table of this section, stating that nothing can be formalized in SysML/Papyrus.

Assessor 2: I agree assessor 1, section skip too.

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	1			-
Coverage of SSRS (D.2.6-X-14.2)	2			-
Traceability to SSRS (D.2.6-X-14.3)	2			-
Extensible to software design (D.2.6-X-16)	3			-
Safety function isolation (D.2.6-X-17)	2			-
Safety properties formalisation (D.2.6-X-22)	1			-
Logical expression (D.2.6-X-28.2.2)	3			-
Timing constraints (D.2.6-X-28.2.3)	1			-
Safety properties validation (D.2.6-X-23.3)	1			-
Logical properties assertion (D.2.6-X-34)	3			-
Proof of assertions (D.2.6-X-34.2)	0			-

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	1			-
Time-outs	0			-
Truth tables	0			-
Arithmetic	0			-
Braking curves	0			-
Logical statements	0			-
Message and fields	0			-

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

D.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

D.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	3	3	3	9
Software architecture description	3	3	2	8
Software constraints	3	2	1	6
Traceability	2	2	2	6
Executable	1	1	*	2*

Assessor 2: Same comments as bellow on executable.

SysMl does not seems adapted to take into account software constraints to define a performing code.

D.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	1	1	1	3
Software architecture description	3	3	3	9
Software constraints	2	1	1	4
Traceability	1	2	2	5
Executable	0	0	*	0*
Conformance to EN50128 § 7.2	3	*	2	5*
Conformance to EN50128 § 7.3	3	*	1	4*
Conformance to EN50128 § 7.4	3	*	1	4*

Assessor 1: I have strictly no experience with norms, so I do not feel able to put a proper evaluation for EN50128 compliance. Since the code is not executable, I do not see the point of putting a criterion for these items anyway.

Assessor 2: According results of the criteria on table A.3 and A.4, SysML alone is not sufficient for a critical software design activity.

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming	0	0	0	0
Fault detection & diagnostic	0	0	0	0
Error detecting code	0	0	0	0
Failure assertion programming	2	2	2	6
Diverse programming	0	0	0	0
Memorising executed cases	0	0	0	0
Software error effect analysis	2	0	0	2
Fully defined interface	3	3	3	9
Modelling	3	3	2	8
Structured methodology	3	2	3	8

D.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	0	0	0	0
Modeling	1	2	1	4
Modular approach (mandatory)	2	2	2	6
Components	1	1	1	3
Design and coding standards (mandatory)	0	0	1	1
Strongly typed programming language	3	3	3	9

D.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3	3	3	9
Automatic translation	1	1	0	2
Code Generation	0	0	0	0
Model verification	0	0	0	0
Test generation	2	2	1	5
Simulation, execution, debugging	0	0	0	0
Formal proof	0	0	0	0

Modelling support

Papyrus provides a graphical editor.

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

Model verification

Which verification on models are provided by the tool?

Test generation

Does the tool allow to generate tests ? For which purpose ?

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ?

Formal proof

Does the tool allow formal proof ? How ?

D.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	3	3	3	9
Portability to operating systems (D2.6-X-37)	1	2	3	6
Cooperation of tools (D2.6-X-38)	3	2	3	8
Robustness (D2.6-X-41)	1	1	1	3
Modularity (D2.6-X-41.1)	3	3	2	8
Documentation management (D.2.6-X-41.2)	2	2	1	5
Distributed software development (D.2.6-X-41.3)	2	2	2	6
Simultaneous multi-users (D.2.6-X-41.4)	2	2	2	6
Issue tracking (D.2.6-X-41.5)	1	2	1	4
Differences between models (D.2.6-X-41.6)	1	2	1	4
Version management (D.2.6-X-41.7)	2	2	1	5
Concurrent version development (D.2.6-X-41.8)	2	2	2	6
Model-based version control (D.2.6-X-41.9)	1	1	1	3
Role traceability (D.2.6-X-41.10)	2	1	1	4
Safety version traceability (D.2.6-X-41.11)	2	2	1	5
Model traceability (D.2.6-01-035)	3	2	2	7
Tool chain integration	3	3	3	9
Scalability	3	2	1	6

D.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3	2	2	7
Proof of correctness (D.2.6-01-42.03)	1	0	0	1
Existing industrial usage	2	2	1	5
Model verification	0	0	0	0
Test generation	2	2	1	5
Simulation, execution, debugging	2	2	1	5
Formal proof	0	0	0	0

Other elements for tool certification

D.11 Other comments

Please to give free comments on the approach.

Assessor 1: *To me the main strengths of Papyrus are*

- *is that its underlying language, SysML, is rather standardized.*
- *its reliance on the Eclipse, and its strong organizational support, which leads to the integration of interesting features with Papyrus, and can foster the integration of additional ones.*

The main weaknesses of Papyrus are:

- *using Papyrus requires a high number of mouse click to perform relatively simple operations: many grop-down lists in the property edition panel in the editor. Modelers might be a bit frustrated by this edition overhead.*
- *it is not very clear what is to be considered as fully formal, and what is to consider as semi-formal.*

Appendix E: SysML with Enterprise Architect

Author Author of the approaches description Cécile Braunstein (Uni. Bremen)

Assessor 1 First assessor of the approaches Uwe Steinke (Siemens)

Assessor 2 Second assessor of the approaches Roberto Kretschmer (TWT)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

E.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name SysML modelisation with Enterprise Architect

Web site `http://www.sparxsystems.com.au/`

Licence Commercial

Abstract

Short abstract on the approach and tool SysML [1] is a graphical language that extends UML for a customize version suitable for system engineering. It may help modeling system within a board range of system variety that may include hardware, software, data, personnel and facilities. It supports the specification, analysis design, verification and validation of complex system.

Enterprise architect (EA) version 9.3 [2] has been used to implement the model. EA provides SysML and UML modelling capabilities. EA is a visual platform for designing and constructing software systems, for business process modeling, and for more generalized modeling purposes. it covers all aspects of the development cycle. The main advantages is the requirement management and tracing, the team work and the include versionning. The main cons : it is not an open source tool.

Publications

Short list of publications on the approach

[1] Object Management Group, *OMG Systems Modeling Language (OMG SysML^{TM})*, www.omgsysml.org, 2012.

[2] Some useful links are given at http://www.sparxsystems.com.au/products/ea/trial.html.

[3] Jon Holt and Simon Perry, *SysML for Systems Engineering,IET, 2008.*

E.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	3	3	3	
Sub-system formal design	3	3	2	
Software design	3	2	3	
Software code generation	1	1	1	

Author: Enterprise architect is able to generate C, java ... code from the UML/SysML representation. This capability has not been tested by me. Moreover from the serialized form of the model (xmi) it is possible to use other tool to perform different task including code gneration.

Assessor 2: Sub-system formal design: SysML is a semiformal language. The syntax is formally defined in the standard, but semantics is expressed in natural language and often purposely left ambigiuos to allow for maximum flexibility of the language. Software design: SysML can be linked in EA to UML model elements since SysML itself is a profile of UML. Therefore the approach is well suited for SW design.

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	2	3	3	
Modeling	3	3	3	
Design	3	2	3	
Code generation	2	1	1	
Verification	1	1	1	
Validation	1	1	2	
Safety analyses	2	2	2	

Author: As said before, the model from enterprise architect may be imported with a bit of effort (adapt parser to EA xmi) into other tools or some tool may be plugged to EA. The model may be then used for the different tasks of the table.

Assessor 2: Documentation: SysML diagrammes can be regarded as a good way of documenting system design. EA can be used to automatically generate documents from the model. Code generation: EA can generate code and allows for cunstamization with templates, but the set of output languages is restricted. To support other output languages substantial modification of EA via plugins is required which is not well documented. Validation: SysML was designed to facilitate the communication between stakeholders with different expertise. The graphical notationis relatively easy to understand and this supports an early validation that customer needs will be satisfied.

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

Plenty of experiences using SysML or UML in embedded system may be found, the reader may refer to the proceeding of international conference such as : Conference on Systems Engineering Research (CSER), International Conference on Model-Based Systems Engineering (MBSE), Embedded realtime software and systems (ERTS).

E.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0	0	0	
Semi-formal language	2	3	3	
Formal language	3	2	2	
Structured language	3	3	3	
Modular language	3	3	3	
Textual language	1	1	1	
Mathematical symbols or code	1	1	3	
Graphical language	3	3	3	

Assessor 2: Mathematical symbols or code: Mathematical code can be inserted in blocks and equation systems can be expressed using parametrics diagrammes.

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	2	2	2	
Simple formalization of properties (D.2.6-X-28.1)	2	2	2	
Scalability : capability to design large model	3	3	3	
Easily translatable to other languages (D.2.6-X-30)	2	2	2	
Executable directly (D.2.6-X-33)	1	1	1	
Executable after translation to a code (D.2.6-X-33)	2	2	2	
(precise if the translation is automatic)	2	2	2	
Simulation, animation (D.2.6-X-33)	1	1	1	
Easily understandable (D.2.6-X-27)	3	2	3	
Expertise level needed (0 High level, 3 few level)	2	2	2	
Standardization (D.2.6-X-29)	3	3	3	
Documented (D.2.6-X-29)	3	3	3	
Extensible language (D.2.6-01-28)	3	3	3	

Assessor 2: Expertise level needed: On the one hand, SysML diagrammes support a large ammount of notation so diagrammes can become quite complex. On the other hand, the graphical notation supports an intuitive understandig of the model.

Documentation

SysML is defined and standardized by OMG. They provide a complete definition of the language and a guideline on the methodology [1]. The serialized form (the textual representation of the graphical model) is also defined and standardized by OMG (see the XMI definition).

Enterprise architect gives tutorial and sample model to start with. The tool is really easy to use.

Language usage

SysML is dedicated on system engineering and suitable for a board range of system variety. Using stereotype and profile the language may be extended and customized according to the user need following the specific domain of the modeled system.

E.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)	3	3	3	
System architecture design (D.2.6-X-10.2)	3	3	3	
System data flow identification (D.2.6-X-10.2.3)	3	3	3	
Sub-system focus (D.2.6-X-10.2.4)	3	3	3	
System interfaces definition (D.2.6-X-10.2.5)	3	3	3	
System requirement allocation (D.2.6-X-10.3)	3	3	3	
Traceability with SRS (D.2.6-X-10.5)	3	3	3	
Traceability with Safety activities (D.2.6-X-11)	3	3	3	

Author: SysML proposes a choice of diagrams to describes the system or the software under consideration. It is then possible to represent the system at different level of abstraction. Moreover one of the main addition of SysML compare to UML is the requirement diagram that makes the link between the model and the requirements.

E.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

E.5.1 Semi-formal model

Concerning semi-formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	3	2	3	
Coverage of SSRS (D.2.6-X-12.2.1)	3	3	3	
Coverage of SSHA (D.2.6-X-12.2.2)	*	*	*	
Management of requirement justification (D.2.6-X-12.2.3)	3	3	3	
Traceability to SSRS (D.2.6-X-12.2.5)	3	3	3	
Traceability of exported requirements (D.2.6-X-12.2.6)	3	3	3	
Simulation or animation (D.2.6-X-13 partial)	2	1	1	
Execution (D.2.6-X-13 partial)	2	2	1	
Extensible to strictly formal model (D.2.6-X-14.3)	3	3	2	
Easy to refine towards strictly formal model (D.2.6-X-14.4)	3	3	2	
Extensible and modular design (D.2.6-X-15)	3	3	3	
Extensible to software architecture and design (D.2.6-X-30)	3	3	3	

Author: The link with the SSRS may be really easy if the SSRS is modeled with SysML.

For the coverage, I do not know and I did not try.

Simulation,animation and execution may be done by additional plug-in or by different tools.

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	1	1	1	
Safety properties formalisation (D.2.6-X-22)	2	2	2	
Logical expression (D.2.6-X-28.2.2)	2	2	2	
Timing constraints (D.2.6-X-28.2.3)	3	3	3	
Safety properties validation (D.2.6-X-23.2)	1	1	1	
Logical properties assertion (D.2.6-X-34)	2	2	1	
Check of assertions (D.2.6-X-34.1)	1	1	1	

Author: SysML formalism may be used to add constraints representing assertion or properties. This can then be check by plug-in or external tools During the modeling activities I did not try these requirements, I can not say much about it.

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3	3	
Time-outs	3	3	3	
Truth tables	1	1	1	
Arithmetic	2	2	2	
Braking curves	2	2	2	
Logical statements	3	3	3	
Message and fields	3	3	3	

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

Author: SysML [3] is a good candidate for theses purposes.

E.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	3	2	3	
Coverage of SSRS (D.2.6-X-14.2)	3	3	3	
Traceability to SSRS (D.2.6-X-14.3)	3	3	3	
Extensible to software design (D.2.6-X-16)	3	2	3	
Safety function isolation (D.2.6-X-17)	*	*	*	
Safety properties formalization (D.2.6-X-22)	3*	3	1	
Logical expression (D.2.6-X-28.2.2)	2	2	3	
Timing constraints (D.2.6-X-28.2.3)	3	3	3	
Safety properties validation (D.2.6-X-23.3)	1	1	1	
Logical properties assertion (D.2.6-X-34)	2	2	1	
Proof of assertions (D.2.6-X-34.2)	1	1	1	

Author:

- *Safety function isolation : I do not know.*
- *Safety properties formalization : I did not try*

Assessor 1:

- *Safety function isolation : No special feature to support this.*

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	3	2	2	
Time-outs	3	3	3	
Truth tables	1	1	1	
Arithmetic	2	2	2	
Braking curves	2	1	1	
Logical statements	3	3	3	
Message and fields	3	3	3	

Author:

The formal semantics of SysML is described in textual form in the UML and SysML standards (see `http://www.omg.org/spec/UML/2.3/Superstructure/PDF/`, `http://www.omg.org/spec/SysML/1.2/PDF/`)

The Chapters about State Machines in the UML document gives a rather clear description (thought not really easy to read ...) about the intended behaviors of state machines and the semantic variation points that are NOT fixed by the standard, but may be interpreted in different ways. More mathematical descriptions can be found in many research papers.

Assessor 1: The ambiguity of state machine semantics let me reduce the "‘state machines"’ criterion to 2.

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

Author: SysML is a good candidate for these purposes

E.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

E.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	3	3	3	
Software architecture description	3	3	3	
Software constraints	3	3	3	
Traceability	3	3	3	
Executable	2	1*	1	

Assessor 1: The EA and SysML approach does not mainly address the design + execution level of software production.

E.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	3	3	3	
Software architecture description	3	3	3	
Software constraints	3	3	3	
Traceability	3	3	3	
Executable	2	1	1	
Conformance to EN50128 § 7.2	3	3	3	
Conformance to EN50128 § 7.3	3	3	3	
Conformance to EN50128 § 7.4	3	3	3	

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming	*	3	*	
Fault detection & diagnostic	*	*	*	
Error detecting code	*	*	*	
Failure assertion programming	3	*	*	
Diverse programming	3	*	*	
Memorising executed cases	*	*	*	
Software error effect analysis	*	*	*	
Fully defined interface	3	3	3	
Modelling	3	3	3	
Structured methodology	3	3	3	

Author: SysML is a modeling language, it can also be used to design software. EA first purpose is not to provide a software development infrastructure. This can be realized through plug-in or with external tools. EA itself does not provide mechanism for error detecting code ...

*Assessor 1: The * criteria are out of scope of EA and SysML; EA und SysML do not inhibit or support it.*

E.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	*	3	2	
Modeling	3	3	3	
Modular approach (mandatory)	3	3	3	
Components	3	3		
Design and coding standards (mandatory)	3	3	1	
Strongly typed programming language	3	3	2	

Assessor 2: Design and coding standards (mandatory): Although SysML is standardized, SysML or EA do not prescribe a certain coding standard, i.e. the language elements can be used freely. This can lead to "‘broken"’ models, e.g. a model containing internal block diagramm without a corresponding block. Coding standards would need to be defined externally. Strongly typed programming language: Blocks can use properties which are not typed. EA does not check types even if they are defined.

E.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3	3	3	
Automatic translation	3	3	2	
Code Generation	2	1	2	
Model verification	1	1	1	
Test generation	1	1	1	
Simulation, execution, debugging	1	1	1	
Formal proof	1	1	1	

Modelling support

Does the tool provide a textual or a graphical editor ? Graphical editor.

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

- XMI
- C++
- Java ...

Model verification

Which verification on models are provided by the tool? None on the basic version of the tool.

Test generation

Does the tool allow to generate tests ? For which purpose ? Not the basic version.

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ? Not the basic version.

Formal proof

Does the tool allow formal proof ? How ? Not the basic version.

E.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	0	0	2	
Portability to operating systems (D2.6-X-37)	2	2	2	
Cooperation of tools (D2.6-X-38)	2	2	2	
Robustness (D2.6-X-41)	3	3	2	
Modularity (D2.6-X-41.1)	3	3	3	
Documentation management (D.2.6-X-41.2)	2	2	2	
Distributed software development (D.2.6-X-41.3)	3	3	2	
Simultaneous multi-users (D.2.6-X-41.4)	1	1	1	
Issue tracking (D.2.6-X-41.5)	2	*	1	
Differences between models (D.2.6-X-41.6)	1	1	1	
Version management (D.2.6-X-41.7)	3	3	3	
Concurrent version development (D.2.6-X-41.8)	1	1	1	
Model-based version control (D.2.6-X-41.9)	0	0	0	
Role traceability (D.2.6-X-41.10)	3	*	*	
Safety version traceability (D.2.6-X-41.11)	0	0	0	
Model traceability (D.2.6-01-035)	3	3	3	
Tool chain integration	2*	2	2	
Scalability	3	3	3	

Author: For the tool chain integration it depends on the integration methods choosen. If it is only by exchanging XMI file, there is no problem.

Assessor 2: Open Source: SysML is a free and open standard. EA is closed source and proprietary.

E.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3	3	3	
Proof of correctness (D.2.6-01-42.03)	2	1	1	
Existing industrial usage	3	3	3	
Model verification	1	1	1	
Test generation	0	0	0	
Simulation, execution, debugging	1	1	1	
Formal proof	0	0	0	

Other elements for tool certification

E.11 Other comments

Please to give free comments on the approach.

Assessor 1: The strength of the SysML/EA approach is the high level transformation of an informal spec into a (semi-)formal one. For executable code generation (including a runnable of the whole functionality in one model), execution, debugging, simulation and verification it should be complemented with languages/tools that focus on these issues.

Assessor 2: I agree to assessor 1.

Appendix F: SCADE

Author Author of the approaches description Uwe Steinke (Siemens)

Assessor 1 First assessor of the approaches David Mentré (MERCE)

Assessor 2 Second assessor of the approaches Cécile Braunstein (Uni. Bremen)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

F.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name: SCADE Suite / SCADE System / SCADE LifeCycle

Web site: `http://esterel-technologies.com`

Licence: Commercial

Abstract

Short abstract on the approach and tool (10 lines max)

SCADE is a formal modelling language targeted for safety-critical embedded control applications in the avionics, rail, automotive and industrial automation domain. SCADE source code can be written as text (for anyone who likes writing plain text) or (more usual) as schematic diagrams.

SCADE models are synchronously clocked data flow and state machines, that can be nested and intermixed with each other without limitations. SCADE provides DO-178B- and EN50128-certified code generators producing C or ADA code as output. SCADE models are therefore concrete, deterministic, executable and verifiable; it allows the production of rapid prototype as well as of safety related target system software.

SCADE comes with an integrated development environment (SCADE Suite IDE) including code generator, graphical simulator, model checker/prover, model test coverage analyzer, report generators, version and requirements management gateway with interfaces to various other tools like static code and timing analyzers, System/SysML modelling tools etc.. The IDE provides automatization interfaces to be controlled from external tools, and all SCADE tools itself can also be used in batch mode. In addition, plugins for Eclipse integration are available.

The SCADE paradigm of synchronously clocked data flow and state machines works perfect for embedded control or industry automation software. It is less suitable for tasks like text processing or computer graphic applications. SCADE models do not only describe the structure of software; instead, they are the software implementation itself too. System architectures typically require a higher abstraction means of description at top level like SysML. While SysML modelling can be achieved with any SysML tools, SCADE System provides an automatic transformation from SysML to SCADE.

Publications

Short list of publications on the approach (5 max)

- `http://www.interested-ip.eu/`
- `http://http://www.interested-ip.eu/final-report.html/`

F.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	1	1	1	3
Sub-system formal design	3	3	3	9
Software design	3	3	3	9
Software code generation	3	3	3	9

Author: SCADE can be used for analyzing tasks on system level, especially to clarify complex system behaviour and functions by practical modelling, execution, simulation and test. For a higher abstraction level, this should be enhanced with system modelling languages as SysML.

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	3	2	3	8
Modeling	3	3	3	9
Design	3	3	3	9
Code generation	3	3	3	9
Verification	3	2	3	8
Validation	3	2	3	8
Safety analyses	0	1	0	1

Assessor 1: Verification capabilities of SCADE Verifier are limited by state space explosion. As far as I know, very few SCADE users are doing formal verification except for very small designs.

The test facilities are however much used.

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

Field of usage: Safety critical systems like

- Rail interlocking systems
- Rail track vacancy detection systems
- Rail train control systems
- Rail Level-crossing protection systems
- Avionic flight controllers

F.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0	0	0	0
Semi-formal language	0	3	0	3
Formal language	3	3	3	9
Structured language	3	3	3	9
Modular language	3	2	3	8
Textual language	3	3	3	9
Mathematical symbols or code	3	3	3	9
Graphical language	3	3	3	9

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	2	2	2	6
Simple formalization of properties (D.2.6-X-28.1)	2	2	2	6
Scalability : capability to design large model	3	2	3	8
Easily translatable to other languages (D.2.6-X-28)	3	3	3	9
Executable directly (D.2.6-X-33)	3	3	3	9
Executable after translation to a code (D.2.6-X-33)	3	3	3	9
(precise if the translation is automatic)	3	3	3	9
Simulation, animation (D.2.6-X-33)	3	3	3	9
Easily understandable (D.2.6-X-27)	3	3	3	9
Expertise level needed (0 High level, 3 few level)	2	2	2	6
Standardization (D.2.6-X-29)	3	0	3	6
Documented (D.2.6-X-29)	3	2	3	8
Extensible language (D.2.6-01-28)	2	2	1	5

Author: SCADE is a strictly textual and graphical formal language. It allows to be extended with user-defined operators. Especially the graphical representation is easy to learn and understand; nevertheless the rich tool suite covering most aspects of a EN50128 compliant process causes an appropriate learning effort by amount.

Assessor 1: I am still skeptical that graphical representation is the best way to represent big systems.

SCADE language lacks some facilities, like library parametrization by parameters (like OCaml functor, C++ template or Ada generics).

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

SCADE provides a detailed documentation set:

- Getting Started

- SCADE Language Tutorial
- SCADE Suite User Manual
- SCADE Suite Technical Manual
- SCADE Suite Libraries Manual
- SCADE Language Primer
- SCADE Language Reference Manual
- Gateway Guidelines for LabView, Rhapsody, Simulink
- RTOS Adaptor Guidelines
- Timing and Stack Analysis Tools
- SCADE LifeCycle Documentation
- SCADE Suite Metamodels
- SCADE Glossary
- ...

Language usage

Describe the possible restriction on the language. SCADE is less suitable for tasks like text processing or computer graphic applications.

F.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.1.1)	3	3	3	9
System architecture design (D.2.6-X-10.1.2)	1	1	0	3
System data flow identification (D.2.6-X-10.2.3)	3	2	3	8
Sub-system focus (D.2.6-X-10.2.4)	2	2	2	6
System interfaces definition (D.2.6-X-10.2.5)	2	3	3	8
System requirement allocation (D.2.6-X-10.3)	3	1	3	7
Traceability with SRS (D.2.6-X-10.5)	3	3	3	9
Traceability with Safety activities (D.2.6-X-11)	3	3	3	9

Author: Although SCADE is not made for system analysis, it can be used for the following aspects on system level: Modelling of (separate) system functions, data flows, state machines and interfaces. It provides an excellent tracebility support between many different kinds of documents and other tools.

F.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

F.5.1 Semi-formal model

Author: SCADE models are formal. Since the following table addresses many aspects that SCADE covers in a formal way it is filled anyway. But keep in mind: it's formal - instead of semi-formal.

Concerning formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	2	2		-
Coverage of SSRS (D.2.6-X-12.2.1)	3	2		-
Coverage of SSHA (D.2.6-X-12.2.2)	3	3		-
Management of requirement justification (D.2.6-X-12.2.3)	3	3		-
Traceability to SSRS (D.2.6-X-12.2.5)	3	3		-
Traceability of exported requirements (D.2.6-X-12.2.6)	3	3		-
Simulation or animation (D.2.6-X-13 partial)	3	3		-
Execution (D.2.6-X-13 partial)	3	3		-
Extensible to strictly formal model (D.2.6-X-14.3)	3	3		-
Easy to refine towards strictly formal model (D.2.6-X-14.4)	3	3		-
Extensible and modular design (D.2.6-X-15)	3	2		-
Extensible to software architecture and design (D.2.6-X-30)	3	3		-

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	2	2		-
Safety properties formalisation (D.2.6-X-22)	2	1		-
Logical expression (D.2.6-X-28.2.2)	3	3		-
Timing constraints (D.2.6-X-28.2.3)	2	2		-
Safety properties validation (D.2.6-X-23.2)	2	2		-
Logical properties assertion (D.2.6-X-34)	2	2		-
Check of assertions (D.2.6-X-34.1)	2	2		-

Author: SCADE is a modelling language for functions. Therefore, only the functional aspects of properties are addressed.

Assessor 1: SCADE lacks facilities to check for example data flow properties (inputs, outputs, in-outs) of a function.

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3		-
Time-outs	3	3		-
Truth tables	3	3		-
Arithmetic	3	3		-
Braking curves	3	3		-
Logical statements	3	3		-
Message and fields	2	2		-

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

Author: SCADE is targeted for these purposes.

F.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	3	3	3	9
Coverage of SSRS (D.2.6-X-14.2)	3	3	3	9
Traceability to SSRS (D.2.6-X-14.3)	3	3	3	9
Extensible to software design (D.2.6-X-16)	3	3	3	9
Safety function isolation (D.2.6-X-17)	3	2	3	8
Safety properties formalisation (D.2.6-X-22)	2	2	2	6
Logical expression (D.2.6-X-28.2.2)	3	3	3	9
Timing constraints (D.2.6-X-28.2.3)	3	2	3	8
Safety properties validation (D.2.6-X-23.3)	2	2	2	6
Logical properties assertion (D.2.6-X-34)	2	2	2	6
Proof of assertions (D.2.6-X-34.2)	2	2	3	7

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3	3	9
Time-outs	3	2	3	8
Truth tables	3	3	3	9
Arithmetic	3	3	3	9
Braking curves	3	3	3	9
Logical statements	3	3	3	9
Message and fields	3	2	3	8

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

Author: SCADE is targeted for these purposes.

Assessor 2: A SCADE model my be directly defined from the SSRS, this implies that code may directly be generated from the SSRS.

F.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

F.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	3	3	3	9
Software architecture description	3	3	3	9
Software constraints	3	2	3	8
Traceability	3	3	3	9
Executable	3	3	3	9

F.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	3	3	3	9
Software architecture description	3	3	3	9
Software constraints	3	3	3	9
Traceability	3	3	3	9
Executable	3	3	3	9
Conformance to EN50128 § 7.2	3	3	3	9
Conformance to EN50128 § 7.3	3	3	3	9
Conformance to EN50128 § 7.4	3	3	3	9

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming	3*	0	3	6
Fault detection & diagnostic	0*	0	0	0
Error detecting code	0*	0	0	0
Failure assertion programming	1	0	0	1
Diverse programming	0*	0	0	0
Memorising executed cases	0*	1	2	3
Software error effect analysis	0*	0	0	0
Fully defined interface	3	3	3	9
Modelling	3	3	3	9
Structured methodology	3	3	3	9

*Author: The * are out of scope of SCADE. It does not support or prohibit these techniques in a specific way.*

F.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	3	2	3	8
Modeling	3	3	3	9
Modular approach (mandatory)	3	3	3	9
Components	3	3	3	9
Design and coding standards (mandatory)	3	2	3	8
Strongly typed programming language	3	3	3	9

Assessor 1: Does SCADE allow to check coding rules at diagram level? I don't think so. Set of properties and extent of their verification is limited in SCADE.

F.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3	3	3	9
Automatic translation	3	3	3	9
Code Generation	3	3	3	9
Model verification	3	2	2	7
Test generation	2	1	1	4
Simulation, execution, debugging	3	3	3	9
Formal proof	3	2	2	7

Modelling support

Does the tool provide a textual or a graphical editor ?

- Yes, both.

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

- Validated translation of SCADE models into C or ADA code.

Model verification

Which verification on models are provided by the tool?

- Simulation
- animation
- verification via manual or script-based test suites
- Model test coverage for structural model coverage measurement
- Formal proving

Test generation

Does the tool allow to generate tests ? For which purpose ?

- SCADE offers test suites to be built via test scripts
- For automatic model based test case generation tools like RT-Tester are applicable

Simulation, execution, debugging

Does the tool allow to simulate or to debug step by step a model or a code ?

- It allows simulation on a clock by clock base by executing the generated code while the model behaviour is visualized graphically
- Graphical model debugging with breakpoint capabilities
- Playback function for logfiles from the field

Formal proof

Does the tool allow formal proof ? How ?

- SCADE integrates the Prover design verifier.
- the provable properties have to be modelled with SCADE Suite and connected to the target model in an target-observer configuration

Assessor 1: Formal verification is limited by state space explosion.

F.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	0	0	0	0
Portability to operating systems (D2.6-X-37)	3*	1	0	4
Cooperation of tools (D2.6-X-38)	3	3	2	8
Robustness (D2.6-X-41)	3	3	3	9
Modularity (D2.6-X-41.1)	3	3	3	9
Documentation management (D.2.6-X-41.2)	3	3	3	9
Distributed software development (D.2.6-X-41.3)	2	2	2	6
Simultaneous multi-users (D.2.6-X-41.4)	1	1	1	3
Issue tracking (D.2.6-X-41.5)	0	0	0	0
Differences between models (D.2.6-X-41.6)	3	3	3	9
Version management (D.2.6-X-41.7)	3	3	3	9
Concurrent version development (D.2.6-X-41.8)	2	2	*	4*
Model-based version control (D.2.6-X-41.9)	3	3	3	9
Role traceability (D.2.6-X-41.10)	0	0	0	0
Safety version traceability (D.2.6-X-41.11)	0	0	0	0
Model traceability (D.2.6-01-035)	3	3	3	9
Tool chain integration	3	3	*	6*
Scalability	3	3	3	9

Author: ** The SCADE tool suite requires MS Windows. The generated executable code is able to run on any operating system and on platforms without an operation system.*

Assessor 2: *managment SCADE is a almost complete tool chain. Missing multi-users and system design.*

F.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

Author: *SCADE is targeted to be certifiable. Validation documentation is available.*

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3	3	3	9
Proof of correctness (D.2.6-01-42.03)	3	3	3	9
Existing industrial usage	3	3	3	9
Model verification	3	2	3	8
Test generation	2	2	2	6
Simulation, execution, debugging	3	3	3	9
Formal proof	3	2	2	7

Other elements for tool certification

F.11 Other comments

Please to give free comments on the approach.

Appendix G: EventB and Rodin

Author Author of the approaches description Matthias Gudemann - Marielle Petit-Doche (Systerel)

Assessor 1 First assessor of the approaches David Mentré (MERCE)

Assessor 2 Second assessor of the approaches Alexander Stante (Fraunhofer)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

G.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name Event-B method and Rodin tool

Web site `http://www.event-b.org/`

Licence open source (EPL licence) available on source forge

Abstract

Short abstract on the approach and tool (10 lines max)

The formal language Event-B is based on a set-theoretic approach. It is a variant of the B language, with a focus on system level modeling and has also be defined by Jean-Raymond Abrial. An Event-B model is separated into a static and a dynamic part.

Rodin is an industrial strength formal modeling tool. It allows the application of the Event-B approach to formal systems modeling. It provides proof obligation creation for invariants, refinement relations and data types. It comprises an Eclipse based modeling framework and supports numerous plugins, e.g., graphical modeling (iUML), automated proof support (theorem provers, SMT solvers) and traceability of requirements (ProR). It was developed by various academic and industrial partners in the European Union Projects RODIN (2003-2007), DEPLOY (2008-2012) and currently ADVANCE (2011-2014).

Publications

Short list of publications on the approach (5 max)

[Abrial2011]Modeling in Event-B: System and Software Engineering

Rodin Handbook (2012): `http://handbook.event-b.org`

[DEPLOY_book2013] Industrial Deployment of System Engineering Methods `http://rodintools.org/flyer.pdf`

G.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	3	3		
Sub-system formal design	3	3		
Software design	2	2		
Software code generation	1	1		

Author: This approach is designed for analyses at the early stage of the development of a system. Its capabilities of abstraction allow easy system level reasoning, without taking into account implementation details. Classical B, based on the same language is better adapted for software design and code generation H.

It is possible to generate C or Ada code from an Event-B model `http://eprints.soton.ac.uk/336226/1/ABZ2012_short_v20120202.pdf` and `http://deploy-eprints.ecs.soton.ac.uk/375/1/AdaEurope2012.pdf`. Two approaches are on development: `http://wiki.event-b.org/index.php/Code_Generation_Activity` and `http://eb2all.loria.fr/`. However, the Event-B specification must be sufficiently detailed to generate code, and this code generator do not take care of software constraints on critical systems or code optimisation.

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	2	2		
Modeling	3	3		
Design	3	3		
Code generation	1	1		
Verification	3	3		
Validation	3	3		
Safety analyses	2	2		

Author: This approach is not sufficient to cover all the safety analyses activities. But it is useful to give confidence on the analyses on a mathematical model. See `http://www.erts2012.org/Site/0P2RUC89/4D-2.pdf`.

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

Author: This approach has already been used in railway on two main critical systems: modelling the DIR41 (instead the generic specification of all interlocking systems used in the metro in Paris) and modelling the track-side controller from the metro of Lyon.

There are others application in the automotive and aeronautic domains. See for more details: `http://wiki.event-b.org/index.php/Industrial_Projects`

G.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0	0		
Semi-formal language	0	0		
Formal language	3	3		
Structured language	3	3		
Modular language	3	2		
Textual language	0	3		
Mathematical symbols or code	3	3		
Graphical language	2	2		

Author: Event-B is a formal language with a mathematical notation. But the Rodin tool can be completed with plug-in like iUML `http://wiki.event-b.org/index.php/`

`IUML-B` allows to model directly state-machine or block diagram graphically and to give the event-B mathematical translation.

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	3	3		
Simple formalization of properties (D.2.6-X-28.1)	3	2		
Scalability : capability to design large model	2	2		
Easily translatable to other languages (D.2.6-X-30)	3	3		
Executable directly (D.2.6-X-33)	0	0		
Executable after translation to a code (D.2.6-X-33) (precise if the translation is automatic)	2	2		
Simulation, animation (D.2.6-X-33)	3	3		
Easily understandable (D.2.6-X-27)	3	2		
Expertise level needed (0 High level, 3 few level)	1	1		
Standardization (D.2.6-X-29)	1	1		
Documented (D.2.6-X-29)	3	3		
Extensible language (D.2.6-01-28)	3	-		

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

Author: A description of the language is available here `http://wiki.event-b.org/index.php/Event-B_Language`.

A complete handbook of the Rodin tool is available here `http://handbook.event-b.org/`.

Currently there is no guidelines or coding rules.

Language usage

Describe the possible restriction on the language

Author: No restriction known.

G.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)	3	3		
System architecture design (D.2.6-X-10.2)	3	3		
System data flow identification (D.2.6-X-10.2.3)	3	2		
Sub-system focus (D.2.6-X-10.2.4)	3	3		
System interfaces definition (D.2.6-X-10.2.5)	3	3		
System requirement allocation (D.2.6-X-10.3)	3	3		
Traceability with SRS (D.2.6-X-10.5)	3	3		
Traceability with Safety activities (D.2.6-X-11)	3	3		

Author: Requirement traceability can be manage with the ProR plug-in `http://wiki.event-b.org/index.php/ProR`.

G.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

G.5.1 Semi-formal model

Author: As a formal language, Event-B can cover some artifacts of semi-formal models.

Concerning semi-formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	2	2		
Coverage of SSRS (D.2.6-X-12.2.1)	2	2		
Coverage of SSHA (D.2.6-X-12.2.2)	2	2		
Management of requirement justification (D.2.6-X-12.2.3)	3	3		
Traceability to SSRS (D.2.6-X-12.2.5)	3	3		
Traceability of exported requirements (D.2.6-X-12.2.6)	3	3		
Simulation or animation (D.2.6-X-13 partial)	3	3		
Execution (D.2.6-X-13 partial)	1	1		
Extensible to strictly formal model (D.2.6-X-14.3)	3	3		
Easy to refine towards strictly formal model (D.2.6-X-14.4)	3	3		
Extensible and modular design (D.2.6-X-15)	3	2		
Extensible to software architecture and design (D.2.6-X-30)	3	3		

Author: Model can be executed after translation.

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	3	3		
Safety properties formalisation (D.2.6-X-22)	3	3		
Logical expression (D.2.6-X-28.2.2)	3	3		
Timing constraints (D.2.6-X-28.2.3)	1	1		
Safety properties validation (D.2.6-X-23.2)	2	3		
Logical properties assertion (D.2.6-X-34)	3	3		
Check of assertions (D.2.6-X-34.1)	3	3		

Author: Concerning time-constraints, it is possible to model abstract properties as time-outs, but not performance constraints.

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3		
Time-outs	2	2		
Truth tables	3	3		
Arithmetic	3	3		
Braking curves	1	1		
Logical statements	3	3		
Message and fields	3	2		

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

Author: Event-B approach is well adapted for system analysis and design. However other formal approach as Classical-B H or Scade are more relevant to be used during software design to code generation of critical systems.

Assessor 1: I'm not sure Event-B is suitable for modelling "message and fields".

G.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	3	3		
Coverage of SSRS (D.2.6-X-14.2)	2	2		
Traceability to SSRS (D.2.6-X-14.3)	3	3		
Extensible to software design (D.2.6-X-16)	3	2		
Safety function isolation (D.2.6-X-17)	3	3		
Safety properties formalisation (D.2.6-X-22)	3	3		
Logical expression (D.2.6-X-28.2.2)	3	3		
Timing constraints (D.2.6-X-28.2.3)	2	1		
Safety properties validation (D.2.6-X-23.3)	2	3		
Logical properties assertion (D.2.6-X-34)	3	3		
Proof of assertions (D.2.6-X-34.2)	3	3		

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3		
Time-outs	2	1		
Truth tables	3	3		
Arithmetic	3	3		
Braking curves	1	1		
Logical statements	3	3		
Message and fields	3	2		

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

Author: Event-B approach is well adapted for system analysis and design. However other formal approach as Classical-B H or Scade are more relevant to be used during software design to code generation of critical systems.

G.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

Author: This section is skipped : for software design classical B is more adapted than event B H.

G.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model				
Software architecture description				
Software constraints				
Traceability				
Executable				

G.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model				
Software architecture description				
Software constraints				
Traceability				
Executable				
Conformance to EN50128 § 7.2				
Conformance to EN50128 § 7.3				
Conformance to EN50128 § 7.4				

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming				
Fault detection & diagnostic				
Error detecting code				
Failure assertion programming				
Diverse programming				
Memorising executed cases				
Software error effect analysis				
Fully defined interface				
Modelling				
Structured methodology				

G.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Author: This section is skipped : for software design classical B is more adapted than event B H.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods				
Modeling				
Modular approach (mandatory)				
Components				
Design and coding standards (mandatory)				
Strongly typed programming language				

G.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3	3		
Automatic translation	3	2		
Code Generation	1	1		
Model verification	3	3		
Test generation	1	1		
Simulation, execution, debugging	3	2		
Formal proof	3	3		

Modelling support

Does the tool provide a textual or a graphical editor ?

Author: The tool provide a textual support, however some plug-ins allow to model graphical diagrams (as UML or SysML diagram) and to give translation in event-B textual language `http://wiki.event-b.org/index.php/IUML-B`

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

Author: Automatic translation are possible from SysML or UML diagram to event-B, and from event-B to C, C++ or Ada code, see `http://wiki.event-b.org/index.php/Code_Generation_Activity` *and* `http://eb2all.loria.fr/`*.*

Model verification

Which verification on models are provided by the tool?

Author:

Syntax, type verification, refinement verification formal proof of properties (validation)

Test generation

Does the tool allow to generate tests ? For which purpose ?

Author:

The tool is not design to generate test, however we can imagine means to generate test from event-B machine (Rodin can be linked to the model checker ProB).

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ?

Author:

Two plug-ins (ProB `http://wiki.event-b.org/index.php/ProB` and animateB `http://wiki.event-b.org/index.php/AnimB`) are currently existing to simulate and animate the Event-B models. It is also possible to animate state machines`http://wiki.event-b.org/index.php/UML-B_-_Statemachine_Animation`.

Formal proof

Does the tool allow formal proof ? How ?

Author:

Formal proof is one of the most important artefacts of the tool. different elements take part of the formal proof:

- *proof generator*
- *interactive prover*
- *automatic prover*
- *plug-in to link to SAT solver*
- *plug-in to model-checker as ProB*
- *plug-in to AtelierB prover*

G.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	3	2		
Portability to operating systems (D2.6-X-37)	3	3		
Cooperation of tools (D2.6-X-38)	3	3		
Robustness (D2.6-X-41)	2	2		
Modularity (D2.6-X-41.1)	3	3		
Documentation management (D.2.6-X-41.2)	3	3		
Distributed software development (D.2.6-X-41.3)	2	2		
Simultaneous multi-users (D.2.6-X-41.4)	1	1		
Issue tracking (D.2.6-X-41.5)	0	0		
Differences between models (D.2.6-X-41.6)	2			
Version management (D.2.6-X-41.7)	3	1		
Concurrent version development (D.2.6-X-41.8)	2	1		
Model-based version control (D.2.6-X-41.9)	2	2		
Role traceability (D.2.6-X-41.10)	1	1		
Safety version traceability (D.2.6-X-41.11)	1	1		
Model traceability (D.2.6-01-035)	3	2		
Tool chain integration	3	3		
Scalability	2	2		

Author: The tool is based on Eclipse on EUPL licence. Thus is can be associated to other tools on eclipse easily, for example: EMF model comparator, Git plug-in,...

G.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3	3		
Proof of correctness (D.2.6-01-42.03)	0	0		
Existing industrial usage	2	2		
Model verification	3	3		
Test generation	0	0		
Simulation, execution, debugging	3	2		
Formal proof	3	3		

Other elements for tool certification

Author: Event-B method is quoted and recommended for system analyses in the draft version of future standard EN50126.

G.11 Other comments

Please to give free comments on the approach.

Appendix H: Classical B and Atelier B

Author Author of the approaches description Marielle Petit-Doche (Systerel)

Assessor 1 First assessor of the approaches Peter Mahlmann (DB)

Assessor 2 Second assessor of the approaches Jan Welte (TU-BS)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

H.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name Classical B and Atelier B

Web site `http://www.atelierb.eu/en/`

Licence Free but close licence (with progressive Open Source implementation of Atelier B tools)

Abstract

The B-Method is a formal method developed by J-R. Abrial and used in industry, especially in railway industry, to develop complex systems. It covers software development from formal specification to code level. Proof mechanisms guaranty the consistency of specifications properties and the complete consistency of code regarding its formal specification. It is efficient to model functional elements of a critical software with respect to EN50128 constraints.

AtelierB is the industrial tool the most used to develop critical software.

Publications

[Abrial1996] The B Book, Assigning Programs to Meanings

H.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	1			
Sub-system formal design	2			
Software design	3			
Software code generation	3			

Author: Classical B can be used for system design, however it is most adapted to software design. Event-B based on the same language is better for system analyses G.

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	0			
Modeling	3			
Design	3			
Code generation	3			
Verification	3			
Validation	2			
Safety analyses	1			

Author: Classical B can be used to validate safety properties.

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

Author:

Classical B has been used successfully in railway industry (mainly by Alstom, Siemens and AREVA) to develop critical software in urban (CBTC, PMI,...) and mainline domains (KVB, Eurobalise,...). Hundred of different systems are running in the world embedding software developed in B (see `http://www.cs.vu.nl/~wanf/pubs/handbookFFM.pdf`, `http://link.springer.com/chapter/10.1007/3-540-48119-2_22` and `http://web.tiscali.it/chiccoterri/MetodB.htm`).

It is also used in nuclear, aeronautic and defence area.

H.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0			
Semi-formal language	0			
Formal language	3			
Structured language	3			
Modular language	3			
Textual language	1			
Mathematical symbols or code	3			
Graphical language	1			

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	3			
Simple formalization of properties (D.2.6-X-28.1)	3			
Scalability : capability to design large model	3			
Easily translatable to other languages (D.2.6-X-30)	3			
Executable directly (D.2.6-X-33)	0			
Executable after translation to a code (D.2.6-X-33) (precise if the translation is automatic)	3			
Simulation, animation (D.2.6-X-33)	3			
Easily understandable (D.2.6-X-27)	2			
Expertise level needed (0 High level, 3 few level)	1			
Standardization (D.2.6-X-29)	3			
Documented (D.2.6-X-29)	3			
Extensible language (D.2.6-01-28)	1			

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

Author:

Language is documented with language manual reference (`http://www.atelierb.eu/ressources/manrefb1.8.6.uk.pdf`), tool with the user manual (`http://www.tools.clearsy.com/resources/User_uk.pdf`). Industrial have developed their own coding rules and guidelines.

Language usage

Describe the possible restriction on the language

Author: Some restrictions in the language manual reference. Industrial guides can add restrictions.

H.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

Acoording WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)	3			
System architecture design (D.2.6-X-10.2)	3			
System data flow identification (D.2.6-X-10.2.3)	3			
Sub-system focus (D.2.6-X-10.2.4)	3			
System interfaces definition (D.2.6-X-10.2.5)	3			
System requirement allocation (D.2.6-X-10.3)	2			
Traceability with SRS (D.2.6-X-10.5)	2			
Traceability with Safety activities (D.2.6-X-11)	2			

Author: Classical B can be used to support partly system analysis. However Event B, based on the same language, is more suitable G.

H.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

H.5.1 Semi-formal model

Author: *As a formal language, classical B can cover some artifacts of semi-formal models.*

Concerning semi-formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	2			
Coverage of SSRS (D.2.6-X-12.2.1)	3			
Coverage of SSHA (D.2.6-X-12.2.2)	2			
Management of requirement justification (D.2.6-X-12.2.3)	1			
Traceability to SSRS (D.2.6-X-12.2.5)	2			
Traceability of exported requirements (D.2.6-X-12.2.6)	1			
Simulation or animation (D.2.6-X-13 partial)	2			
Execution (D.2.6-X-13 partial)	1			
Extensible to strictly formal model (D.2.6-X-14.3)	3			
Easy to refine towards strictly formal model (D.2.6-X-14.4)	3			
Extensible and modular design (D.2.6-X-15)	3			
Extensible to software architecture and design (D.2.6-X-30)	3			

Author: *Execution is possible after translation.*

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	3			
Safety properties formalisation (D.2.6-X-22)	2			
Logical expression (D.2.6-X-28.2.2)	3			
Timing constraints (D.2.6-X-28.2.3)	1			
Safety properties validation (D.2.6-X-23.2)	3			
Logical properties assertion (D.2.6-X-34)	3			
Check of assertions (D.2.6-X-34.1)	3			

Author: *Timing constraints like time-outs can be modelled.*

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3			
Time-outs	2			
Truth tables	3			
Arithmetic	3			
Braking curves	2			
Logical statements	3			
Message and fields	3			

Author: Braking curves have already been specifyed in classical b in past industrial projects. However this need specific skills.

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

Author: Classical B approach is well adapted for software design untill code generation. However a formal approach as Event B G is more relevant to be used for system analysis and design.

H.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	3			
Coverage of SSRS (D.2.6-X-14.2)	3			
Traceability to SSRS (D.2.6-X-14.3)	2			
Extensible to software design (D.2.6-X-16)	3			
Safety function isolation (D.2.6-X-17)	3			
Safety properties formalisation (D.2.6-X-22)	2			
Logical expression (D.2.6-X-28.2.2)	3			
Timing constraints (D.2.6-X-28.2.3)	2			
Safety properties validation (D.2.6-X-23.3)	3			
Logical properties assertion (D.2.6-X-34)	3			
Proof of assertions (D.2.6-X-34.2)	3			

Author: Timing constraints like time-outs can be modelled.

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	3			
Time-outs	2			
Truth tables	3			
Arithmetic	3			
Braking curves	2			
Logical statements	3			
Message and fields	3			

Author: Braking curves have already been specifyed in classical b in past industrial projects. However this need specific skills.

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

Author: Classical B approach is well adapted for software design untill code generation. However a formal approach as Event B G is more relevant to be used for system analysis and design.

H.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

H.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	3			
Software architecture description	3			
Software constraints	3			
Traceability	3			
Executable	3			

H.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	3			
Software architecture description	3			
Software constraints	3			
Traceability	3			
Executable	3			
Conformance to EN50128 § 7.2	3			
Conformance to EN50128 § 7.3	3			
Conformance to EN50128 § 7.4	3			

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming	3			
Fault detection & diagnostic	1			
Error detecting code	1			
Failure assertion programming	2			
Diverse programming	0			
Memorising executed cases	0			
Software error effect analysis	0			
Fully defined interface	3			
Modelling	3			
Structured methodology	3			

H.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	3			
Modeling	3			
Modular approach (mandatory)	3			
Components	3			
Design and coding standards (mandatory)	3			
Strongly typed programming language	3			

H.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3			
Automatic translation	3			
Code Generation	3			
Model verification	3			
Test generation	0			
Simulation, execution, debugging	2			
Formal proof	3			

Modelling support

Does the tool provide a textual or a graphical editor ?

Author: Textual editor

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

Author: Automatic translation to C, Ada or HIA is possible with existing tools. Automatic translators to other languages can be developed.

Model verification

Which verification on models are provided by the tool?

Author: simulation, verification, validation and formal proof, test coverage

Test generation

Does the tool allow to generate tests ? For which purpose ?

Author: No

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ?

Author: Code can be simulated step by step with specific tool of the target language or animated with ProB `http://www.tools.clearsy.com/wp1/?page_id=124`.

Formal proof

Does the tool allow formal proof ? How ?

Author: Yes, a set of rules describes how to produce proof obligations to cover verification of the model (type verification, invariant preservation, refinement,...) A set of rules can be defined to write proofs.

H.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	1			
Portability to operating systems (D2.6-X-37)	3			
Cooperation of tools (D2.6-X-38)	2			
Robustness (D2.6-X-41)	3			
Modularity (D2.6-X-41.1)	3			
Documentation management (D.2.6-X-41.2)	3			
Distributed software development (D.2.6-X-41.3)	2			
Simultaneous multi-users (D.2.6-X-41.4)	2			
Issue tracking (D.2.6-X-41.5)	0			
Differences between models (D.2.6-X-41.6)	2			
Version management (D.2.6-X-41.7)	1			
Concurrent version development (D.2.6-X-41.8)	1			
Model-based version control (D.2.6-X-41.9)	0			
Role traceability (D.2.6-X-41.10)	1			
Safety version traceability (D.2.6-X-41.11)	0			
Model traceability (D.2.6-01-035)	2			
Tool chain integration	2			
Scalability	3			

Author: The tool is partly open-source, but it is free for use. For more details `http://www.atelierb.eu/outil-atelier-b/`.

H.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3			
Proof of correctness (D.2.6-01-42.03)	3			
Existing industrial usage	3			
Model verification	3			
Test generation	0			
Simulation, execution, debugging	3			
Formal proof	3			

Other elements for tool certification

Author: *Method and tool have already been used to develop numbers of SIL4 software for railway.*

H.11 Other comments

Please to give free comments on the approach.

Appendix I: Petri Nets

Author Author of the approaches description Jan Welte (TU-BS)

Assessor 1 First assessor of the approaches Marc Antoni (SNCF)

Assessor 2 Second assessor of the approaches Cyril Cornu (All4Tec)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, these have been given using the following list:

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

Where needed additional comments for the given notes are under each table.

I.1 Presentation

This section gives a quick presentation of the approach and the supporting tool.

Name means of description: Coloures Petri nets; tool: CPN Tools

Web site http://cpntools.org

Licence CPN Tools GUI: GNU GPL version 2; Simulator: GNU GPL version 2 andBSD license; Access/CPN: GNU LGPL version 2.1

Abstract

The following evaluation of petri nets and the CPN Tools is in contrast to all other means of description not based on a preliminary model but on earlier formal model of the then current ERTMS/ETCS specification done for Deutsche Bahn AG in 1997. This work developed a full formal system model which was able to provide a universal means of description for all the development phases. Thereby, an integrated event and data-oriented approach has been used, which is able to describe the different aspects of the system on their own net levels. Coloured Petri nets have been as means of description for this procedure, as they permit universal application, the use of different methods and formal analysis. CPN Tools is a mature tool suite which provides support to edit, check, simulate and analyse nets on all relevant abstraction levels.

Publications

- Van der Aalst, W.M.P. and Stahl, C.: Modeling Business Processes – A Petri Net-Oriented Approach. The MIT Press, 2011.
- K. Jensen and Kristensen, L.M.: Coloured Petri Nets – Modeling and Validation of Concurrent Systems. Springer-Verlag Berlin, 2009.
- Janhsen, A.; Lemmer, K.; Ptok, B.; Schnieder, E.: Formal Specifications of the European Train Control System. In: Papageorgiou, M.; Pouliezos, A., Hrsg.: Proceedings of 8th IFAC Symposium on Transportation Systems, S. 1215-1220, Chania, Juni 1997. Chania.
- Meyer zu Hörste, M.; Ptok, B.; Schnieder, E.; Schulz, H.-M.: Modelling and Simulation of the European Train Control System for test case generation. In: Mellitt, B; Hill, R. J.; Allan, J.; Sciutto, G.; Brebbia, C. A., Hrsg.: Proceedings COMPRAIL '98: Computers in Railways VI, Lissabon, S. 649-658, Southampton, Boston, Juli 1998. Sixth international conference on computer aided design, manufacture and operation in the railway and other advanced mass transit systems / Lissabon, WIT Press.
- Jensen, K.; Kristensen, L.M. and Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer (STTT)9(3-4), pp. 213-254, 2007.

I.2 Main usage of the approach

The approach has been used to develop a strictly formal model based on an earlier System Requirements Specification version. Therefore the approach uses context, process, scenario and function models to provide visualisation of the system and sub-systen processes as well as the operational processes. Petri Nets as a means of description are used in research an in industrial applications used for Process Modeling, Data analysis, Software design and Reliability engineering. Coloured Petri Nets are High-level Petri Nets which are mainly used to describe, simulate and validated communication between humans and/or computers. As a means of description Coloured and Hierarchic Petri nets allow to use one uniform means of description for the entire development cycle, starting with the specification through to implementation. In addition Petri nets provide the capacity that allows different methods to be used during one single phase of the development cycle and also phase-specific methods. CPN Tools has a graphical editor to model nets and provides various methods to analyse the nets, most importantly a reachability analysis. Resprechtivily the tool supports most applications for Coloured Petri Nets.

According to the figure 1, the approach is recommanded for the following phases (note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	3		3	
Sub-system formal design	3		3	
Software design	3		3	
Software code generation	1		1	

According to the figure 1, the approach is recommanded for the following types of activities (note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	1		1	
Modeling	3		3	
Design	3		2	
Code generation	1		1	
Verification	3		3	
Validation	3		3	
Safety analyses	3		2	

Known usages

The first goal of the openETCS project, formalising subset 26, has already been handled by this approach as an earlier version of the ERTMS system requirement specifications has been modeled this way. Thereby it was demonstrated that during the phases of system development, covering the system specification through to the final system design, a model based on Petri nets can be used.

I.3 Language

This section discusses the main element of the language.

Coloured Petri nets have the following main characteristics:

	Author	Assessor 1	Assessor 2	Total
Informal language				
Semi-formal language				
Formal language	x			
Structured language	x			
Modular language	x			
Textual language				
Mathematical symbols or code	x			
Graphical language	x			

According WP2 requirements, notes are given for the capabilities of Coloured Petri nets (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	3		3	
Simple formalization of properties (D.2.6-X-28.1)	3		3	
Scalability : capability to design large model	3		3	
Easily translatable to other languages (D.2.6-X-30)	2		2	
Executable directly (D.2.6-X-33)	3		3	
Executable after translation to a code (D.2.6-X-33) (precise if the translation is automatic)	3 2		3 2	
Simulation, animation (D.2.6-X-33)	3		3	
Easily understandable (D.2.6-X-27)	2		2	
Expertise level needed (0 High level, 3 few level)	1		1	
Standardization (D.2.6-X-29)	3		3	
Documented (D.2.6-X-29)	3		3	
Extensible language (D.2.6-01-28)	3		3	

Documentation

Coloured petri nets are standardised as part of the high level petri nets in ISO/IEC 15909 Systems and software engineering - High-level Petri nets. The use of petri nets for the system dependability analysis is standardised in IEC 62551 Analysis techniques for dependability - Petri net modeling. In addition Coloured petri nets and the CPN Tools are introduced and documented in the book *Coloured Petri Nets – Modeling and Validation of Concurrent Systems* by K. Jensen and L.M. Kristensen.

Language usage

Basically petri nets are a very powerful completely mathematical defined means of description to graphically model systems in a discrete way. Over the time various extensions have be developed to extend the concept of petri nets to handle a larger amount of system properties and behaviour. Some of these extensions require export knowledge, so that their models can be difficult to understand for users not familiar with the theory behind petri nets.

I.4 System Analysis

This section discusses the usage of the approach for system analysis.

Acoording WP2 requirements, the approach can be involved for the sub-system requirement specification in the following ways:

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)	3		3	
System architecture design (D.2.6-X-10.2)	3		3	
System data flow identification (D.2.6-X-10.2.3)	3		3	
Sub-system focus (D.2.6-X-10.2.4)	3		3	
System interfaces definition (D.2.6-X-10.2.5)	3		3	
System requirement allocation (D.2.6-X-10.3)	2		2	
Traceability with SRS (D.2.6-X-10.5)	2		2	
Traceability with Safety activities (D.2.6-X-11)	2		2	

I.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design.

Petri nets can be used to model both kind of models during this phase: semi-formal models tocover the SSRS (D.2.6-X-12.1) and strictly formal models to focus on some functional and safety aspects (D.2.6-X-14). Obviously petri nets as a strictly formal means can be used to define the semi-formalmodel.

I.5.1 Semi-formal model

Concerning semi-formal model, the following WP2 requirements are covered:

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	2		*	
Coverage of SSRS (D.2.6-X-12.2.1)	3		*	
Coverage of SSHA (D.2.6-X-12.2.2)	3		2	
Management of requirement justification (D.2.6-X-12.2.3)	1		1	
Traceability to SSRS (D.2.6-X-12.2.5)	2		2	
Traceability of exported requirements (D.2.6-X-12.2.6)	2		2	
Simulation or animation (D.2.6-X-13 partial)	3		3	
Execution (D.2.6-X-13 partial)	3		3	
Extensible to strictly formal model (D.2.6-X-14.3)	3		3	
Easy to refine towards strictly formal model (D.2.6-X-14.4)	3		3	
Extensible and modular design (D.2.6-X-15)	3		3	
Extensible to software architecture and design (D.2.6-X-30)	3		3	

*Assessor 2: * Since SSRS is not finished at the moment, those criteria can't be evaluated*

Concerning safety properties management, the following WP2 requirements are covere:

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	3		3	
Safety properties formalisation (D.2.6-X-22)	3		3	
Logical expression (D.2.6-X-28.2.2)	3		3	
Timing constraints (D.2.6-X-28.2.3)	3		3	
Safety properties validation (D.2.6-X-23.2)	3		3	
Logical properties assertion (D.2.6-X-34)	3		3	
Check of assertions (D.2.6-X-34.1)	3		3	

Coloured Petri nets allow to formalize the following(D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3		3	
Time-outs	3		3	
Truth tables	2		2	
Arithmetic	3		3	
Braking curves	1		1	
Logical statements	3		3	
Message and fields	3		3	

Additional comments on semi-formal model

Since the approach has already been used to model the ERTMS/ETCS specification in an earlier version, it is sufficient for the task. As petri nets are closely related to all means of descriptions based on state machines and automatas the can be compared or translated relatively easy. In addition all verification and validation activities supported by those means of descriptions can be used on petri nets. Although hazard and risk analysis techniques as FTA and FMEA can be translated intro petri nets and petri nets can be used for formal hazard and risk analysis methods.

I.5.2 Strictly formal model

Concerning the strictly formal model, the following WP2 requirements are covered:

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	3		3	
Coverage of SSRS (D.2.6-X-14.2)	3		*	
Traceability to SSRS (D.2.6-X-14.3)	2		*	
Extensible to software design (D.2.6-X-16)	3		3	
Safety function isolation (D.2.6-X-17)	3		3	
Safety properties formalisation (D.2.6-X-22)	3		3	
Logical expression (D.2.6-X-28.2.2)	3		3	
Timing constraints (D.2.6-X-28.2.3)	3		3	
Safety properties validation (D.2.6-X-23.3)	3		3	
Logical properties assertion (D.2.6-X-34)	3		3	
Proof of assertions (D.2.6-X-34.2)	3		3	

*Assessor 2: * Since SSRS is not finished at the moment, those criteria can't be evaluated*

Coloured petri nets allow to formalize the following(D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	3		3	
Time-outs	3		3	
Truth tables	2		2	
Arithmetic	3		3	
Braking curves	1		1	
Logical statements	3		3	
Message and fields	3		3	

Additional comments on semi-formal model

Petri nets models can be build directly from the SSRS and then be systematically refined. Since petri nets are closely related to all means of descriptions based on state machines and automatas the can be compared or translated relatively easy. In addition all verification and validation activities supported by those means of descriptions can be used on petri nets. Although hazard and risk analysis techniques as FTA and FMEA can be translated intro petri nets and petri nets can be used for formal hazard and risk analysis methods.

I.6 Software design

This section discusses the usage of the approach for software design.

I.6.1 Functional design

The approach allows to produce a functional software model of the on-board unit in the following ways:

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	3		3	
Software architecture description	3		3	
Software constraints	3		3	
Traceability	2		2	
Executable	3		3	

Since petri nets are a formal means of description usually all refinements require a formal model.

I.6.2 SSIL4 design

The approach allows to produce in safety a software model by the following ways:

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	3		3	
Software architecture description	3		3	
Software constraints	3		3	
Traceability	2		2	
Executable	3		3	
Conformance to EN50128 § 7.2	3		3	
Conformance to EN50128 § 7.3	3		3	
Conformance to EN50128 § 7.4	3		3	

The following criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming				
Fault detection & diagnostic	2		2	
Error detecting code	1		1	
Failure assertion programming				
Diverse programming	1		1	
Memorising executed cases	1		1	
Software error effect analysis	3		3	
Fully defined interface	3		3	
Modelling	3			
Structured methodology	3		3	

I.7 Software code generation

CPN Tools does not directly generate source code, but the petri nets models are provided in the CPN ML file format, which can be used by other tools to create code

The following criteria for software design and implementation are covered by the methodology: (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	3		3	
Modeling	3		3	
Modular approach (mandatory)	3		3	
Components	3		3	
Design and coding standards (mandatory)	2		2	
Strongly typed programming language	0		0	

I.8 Main usage of the tool

This section discusses the main usage of the tool.

The following tasks are covered by the tool:

	Author	Assessor 1	Assessor 2	Total
Modelling support	3		3	
Automatic translation	1		1	
Code Generation	1		1	
Model verification	3		3	
Test generation	2		1	
Simulation, execution, debugging	3		3	
Formal proof	3		3	

Modelling support

Since petri nets are a primarily graphic means of descrition CPN Tools provide a graphical modelling editor.

Automatic translation and code generation

CPN Tools does not directly generate source code, but the petri nets models are provided in the CPN ML file format, which can be used by other tools to create code. Petri nets can relatively easy been translated to other means of description based on state machines and automatas.

Model verification

Petri nets are mainly verified by generation and analysis of the state space. The tool supports the calculation and drawing of the state space, which is used to verify certain logical and temporal properties of the system.

Test generation

CPN Tools itself allows simulation of the models. It does not support test generation, but provides interfaces for other tools to do so. Correspondingly, tools like SPENAT can be used to generate and manage all kinds of tests for the nets created with CPN Tools.

Simulation, execution, debugging

The simulation engine of CPN tools provides a powerful simulation of petri nets and has a number of debugging functions.

Formal proof

Petri nets are a strictly formal means of description suited for formal proof of behavioural and structural properties. The analysis of the state space can provide proofs for some kinds of properties. Additional model checker can be combined with the tool to provide aditional functionalities.

I.9 Use of the tool

According WP2 requirements, the following notes are given on characteristics of the use of the tool (from 0 to 3):

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	3 (almost complete)		3	
Portability to operating systems (D2.6-X-37)	3		3	
Cooperation of tools (D2.6-X-38)	2		2	
Robustness (D2.6-X-41)	2		2	
Modularity (D2.6-X-41.1)	3		3	
Documentation management (D.2.6-X-41.2)	0		0	
Distributed software development (D.2.6-X-41.3)	2		2	
Simultaneous multi-users (D.2.6-X-41.4)	2		*	
Issue tracking (D.2.6-X-41.5)	0		0	
Differences between models (D.2.6-X-41.6)	1		1	
Version management (D.2.6-X-41.7)	0		0	
Concurrent version development (D.2.6-X-41.8)	0		0	
Model-based version control (D.2.6-X-41.9)	1		1	
Role traceability (D.2.6-X-41.10)	0		0	
Safety version traceability (D.2.6-X-41.11)	0		0	
Model traceability (D.2.6-01-035)	2		2	
Tool chain integration	2		2	
Scalability	3		3	

I.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3		3	
Proof of correctness (D.2.6-01-42.03)	3		3	
Existing industrial usage	2		2	
Model verification	3		3	
Test generation	1		1	
Simulation, execution, debugging	3		3	
Formal proof	2		2	

Other elements for tool certification

This issue can not be specified at this point.

I.11 Other comments

In the context of this approach coloured petri nets present only a very well suited formal means of description. However the method is basically independent of the means of description and can also be applied on other formal means of description. Thereby the proven method to build context models which are then further refined in process, scenario and function models can be successfully used for many formal means of description.

Appendix J: System C

Author Author of the approaches description Stefan Rieger (TWT)/ Frank Golatowski (Uni Rostock)

Assessor 1 First assessor of the approaches Cecile Braunstein (Uni. Bremen)

Assessor 2 Second assessor of the approaches Silvano Dal Zilio / LAAS

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

J.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name SystemC

Web site `http://www.accellera.org/downloads/standards/systemc/about_systemc/`

Licence SystemC Open Source License

Abstract

SystemC is a C++ library providing an event-driven simulation interface suitable for electronic system level design. It enables a system designer to simulate concurrent processes. SystemC processes can communicate in a simulated real-time environment, using channels of different datatypes (all C++ types and user defined types are supported). SystemC supports hardware and software synthesis (with the corresponding tools). SystemC models are executable.

Publications

Short list of publications on the approach (5 max)

- D. C. Black, SystemC: From the ground up. Springer, 2010.
- IEEE 1666 Standard SystemC Language Reference Manual, `http://standards.ieee.org/getieee/1666/`
- The ITEA MARTES Project, from UML to SystemC, `http://www.martes-itea.org/`
- J. Bhasker, A SystemC Primer, Second Edition, Star Galaxy Publishing, 2004
- F. Ghenassia (Editor), Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems, Springer 2006

J.2 Main usage of the approach

SystemC is suitable for system level design at various abstraction levels (from high level down to individual hardware components) and can thus be employed to build a full system model. Due to its modular design and abstraction principles sub-components and a lower abstraction level of the model can be considered as "black boxes". SystemC models can be executed and simulated allowing for testing of the entire model or individual components.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	1	0	1	2
Sub-system formal design	3	*	1	4*
Software design	3	3	3	9
Software code generation	2	2**	2	6

Assessor 1:

(*) How is it formal ?

Author: The question is whether you consider only fully *formal or also* semi *formal modelling methods here. As there is no separate item for semi formal sub-system design I considered it to be included.*

(**) Is it the SysML to SystemC code generation you have in mind here ?

Author: No, the generation of low-level software from the SystemC model.

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	0	0	0	0
Modeling	3	3	2	8
Design	3	3	3	9
Code generation	3	*	*	3*
Verification	2	*	**	2*
Validation	2	3	2	7
Safety analysis	2	*	**	2*

Assessor 1: This is not explained in the report or shown in the provided example.

Assessor 2:

(*) One of the problem here is that simulation of SystemC models do not take into account the execution platform, like for example the actual hardware limitation of the EVC or the characteristics of the API. How precisely should you model the hardware part of the system ?

> *Author: If you would model the hardware in SystemC, you could have a quite exact model of the entire system. This, however, is not in the scope of openETCS and would increase the modelling effort tremendously. Which other approach takes into account the actual behaviour of the hardware? I don't see that this will be done in any of the approaches as we model only the software. The integration of the API should be doable.*

(**) Simulation and testing can be used for V&V, with the usual caveats associated with non-formal languages. I also agree with Assessor 1 on the fact that safety and verification capabilities are not really demonstrated right now. Actually I am quite doubtful about safety since the only type of safety analysis that I can see performed is error injection during the simulation.

> *Author: You are right, there is no verification integrated into the current state of the model. This is due to time constraints. Of course, you will not be able to fully formally verify a SystemC model in general. You will have to use testing approaches.*

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ?

SystemC has been applied, among others, in the following areas:

- Communication technology
- Hardware design and simulation
- Hardware and software synthesis
- Sensor circuits

- Automotive
- Aerospace industry

SystemC is widely employed in industry. Among the members of the Accellera Systems Initiative responsible for SystemC are the following organisations:

AMD, ARM, Cadence, Intel, NXP, Qualcomm, Synopsys, Texas Instruments, Altera, Boeing, Cisco, Ericsson, Fraunhofer IIS, IBM, NEC, nVidia, Xilinx

Vendors supporting SystemC (according to Wikipedia):

Aldec, AutoESL, Cadence Design Systems, HCL Technologies, Calypto, CircuitSutra, CoFluent Design, CoSynth Synthesizer, CoWare, Forte Design Systems, Mentor Graphics, OVPsim, NEC CyberWorkBench, Imperas, Synopsys, SystemCrafter, JEDA Technologies, HIFSuite, Dynalith Systems, VWorks

J.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0	0	0	0
Semi-formal language	3	3	3	9
Formal language	2	1	1	4
Structured language	3	3	3	9
Modular language	3	3	3	9
Textual language	3	3	3	9
Mathematical symbols or code	2*	2*	2	6
Graphical language	**	0**	0	0*

*Author: * We would set this to 2, because there are languages where you can write the mathematical formula directly like in written text. ** Not graphical, but we are investigating SystemC and UML/SysML integration, the ITEA MARTES Project is addressing this aspect*

Assessor 1:

(*) But it is possible to use a variety of C++ library

Author: True.

(**) SystemC itself is not a graphical language

Author: True as well. We never stated anything else.

According to WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	1	1	1	3
Simple formalization of properties (D.2.6-X-28.1)	1	1	1	3
Scalability : capability to design large model	3	3	3	9
Easily translatable to other languages (D.2.6-X-30)	2	2	2*	6
Executable directly (D.2.6-X-33)	3	3	*	6*
Executable after translation to a code (D.2.6-X-33)	3	3	3	9
(precise if the translation is automatic)			2*	2*
Simulation, animation (D.2.6-X-33)	3*	3*	3	9
Easily understandable (D.2.6-X-27)	2	2	3	7
Expertise level needed (0 High level, 3 few level)	1	2	1	4
Standardization (D.2.6-X-29)	3	3	3**	9
Documented (D.2.6-X-29)	3	3	3	9
Extensible language (D.2.6-01-28)	3	3	3	9

Author: ** Should be two rows. SystemC provides simulation but not animation.*

Assessor 1: ** Some library may be used to trace the value of the variables from simulation. his is not a code animation but this offer another view of the model behaviors.*

Assessor 2:

(*) There is a large collection of tools providing translation from and to SystemC, so I am even willing to put a 3 for "translatability". Once again, to the best of my knowledge, there are no "certified compiler" for SystemC, like it is the case for SCADE for example, so it is necessary to check the result of the translation.

(**) It is not clear if (or where) all the proprietary tools follow the standards consistently. By definition, Accellera provides an implementation of the SystemC standard

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

There is an IEEE Standard covering SystemC, an official specification from the Accellera Initiative and a plethora of third party literature and tutorials.

Language usage

Describe the possible restriction on the language

- As the language is based on C++ and thus inherits its expressivity there might be problems in static analysis if the models use the power of the language in an unrestricted manner.
- The language is text-based and not graphical. However, there are approaches of integrating SystemC and UML/SysML. We are currently investigating in this issue.

J.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

According WP2 requirements, how the approach can be involved for the sub-system requirement specification ?

Author: This is not the main purpose of SystemC. Thus we skip this section.

J.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

J.5.1 Semi-formal model

Concerning semi-formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	3	3	3	9
Coverage of SSRS (D.2.6-X-12.2.1)	*	2	*	2*
Coverage of SSHA (D.2.6-X-12.2.2)	*	1	*	1*
Management of requirement justification (D.2.6-X-12.2.3)	2	2	2	6
Traceability to SSRS (D.2.6-X-12.2.5)	**	2**	*	2*
Traceability of exported requirements (D.2.6-X-12.2.6)	***	2**	*	2*
Simulation or animation (D.2.6-X-13 partial)	3****	3	3	9
Execution (D.2.6-X-13 partial)	3	3	3	9
Extensible to strictly formal model (D.2.6-X-14.3)	2	1	1	4
Easy to refine towards strictly formal model (D.2.6-X-14.4)	2	2	2	6
Extensible and modular design (D.2.6-X-15)	3	3	3	9
Extensible to software architecture and design (D.2.6-X-30)	3	3	3	9

*Author: * The coverage has to be achieved by the* ***model****, not by the language or tool and should be removed from the table.*
*** See table above*
**** What are "exported requirements"?*
***** This cannot be extracted from the requirement as the requirement is on execution which is in the next row.*

Assessor 1:

* *The coverage may just be a metric, the number of function implemented versus the total ETCS functions. I think this can be easily added*

** *This may be achieve by standardized comments.*

Assessor 2:

* *I am not familiar enough with the SSRS and SSHA to provide a valid answer at the moment.*

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	*	*	*	*
Safety properties formalisation (D.2.6-X-22)	2	1*	*	3*
Logical expression (D.2.6-X-28.2.2)	3	3	3	9
Timing constraints (D.2.6-X-28.2.3)	3	3	3	9
Safety properties validation (D.2.6-X-23.2)	3	3	3	9
Logical properties assertion (D.2.6-X-34)	3	3	3	9
Check of assertions (D.2.6-X-34.1)	3	3	3	9

Author: * *Item not clear to me, should be a requirement for the actual implementation, not a model*

Assessor 1: *(*) The safety analysis may be done with external temporal model checker for example or by means of monitor.*

> *Author:* *There are approaches for model checking SystemC models, such as transformations to the timed model checker UPPAAL. However, as far as I know there are no ready-to-use (free) tools.*

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3	3	2*	8
Time-outs	3	3	3	9
Truth tables	3	3	3	9
Arithmetic	3	3	3	9
Braking curves	3	3	3	9
Logical statements	3	3	3	9
Message and fields	3	3	3	9

Assessor 2:

* *State machines are not part of the language and should be "interpreted" in the code.*

Author: I am not exactly sure what is meant with "interpreted" here. In our model we provided an example how you can transform state machines, e.g., SysML statecharts, to SystemC (or any other procedural programming language) in a structured manner.

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

J.5.2 Strictly formal model

Author: Not filled, since we do not consider SystemC to be a strictly formal modelling language, as it has no mathematically formalized sematics. Fully formal models should also support "really" formal verification (not only testing) which requires additional work here. However, there are many approaches in the literature to, e.g., apply model checking to SystemC models.

Additional comments on strictly formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

J.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

Author: SystemC allows system, software and hardware design and is thus suitable.

J.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	*	2	2	4*
Software architecture description	3	3	3*	9
Software constraints	3	3	3*	9
Traceability	2**	2	2*	6
Executable	3	3	3	9

*Author: * Derivation not necessary because the model is fully executable*
*** Can possibly be covered by an associated SysML model. In addition, standardised, machine readable comments in the code could be used.*

Assessor 2:

* *The architecture description cannot be easily extracted from the source files, it is spread around the source code; this is not a problem if SystemC is only use as the target of transformations though.*

Author: True. That's why we advocate a SysML-based approach generating (parts) of the SystemC model from a more abstract SysML model.

J.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	*	*	2*	2*
Software architecture description	3	3	3	9
Software constraints	3	3	3	9
Traceability	2**	2	2	6
Executable	3	3	3	9
Conformance to EN50128 § 7.2	***	0	*	0*
Conformance to EN50128 § 7.3	***	3	*	3*
Conformance to EN50128 § 7.4	***	3	*	3*

Assessor 1:

- *EN50128 § 7.2 : Describe complete set of requirements for the software + describe the overall software test specification*
- *EN50128 § 7.3 : Develop a software architecture*
- *EN50128 § 7.4 : Develop a software component design*

*Author: * Item unclear to me*
*** Can possibly be covered by an associated SysML model. In addition, standardised, machine readable comments in the code could be used.*
**** No idea, why don't you cite these items?*

Assessor 2:

* *The approach followed in the demonstrator is based on Model-Driven Engineering (with the MOF model to text transform) and the use of the Acceleo Eclipse plugin.*

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming	*	*	*	*
Fault detection & diagnostic	2	2	2	6
Error detecting code	3	3	3	9
Failure assertion programming	3	3	3	9
Diverse programming	*	*	*	*
Memorising executed cases	3	3	3	9
Software error effect analysis	*	2*	*	2*
Fully defined interface	3	3	3	9
Modelling	3	*	2	5*
Structured methodology	3	*	*	3*

Author: * *SystemC is a language and no methodology. These methodologies can be applied for most languages.*

Assessor 1: * *One can create mutant.*

J.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	0*	0	0	0
Modeling	3	3	3	9
Modular approach (mandatory)	3	3	3	9
Components	3	3	3	9
Design and coding standards (mandatory)	**	3	3	6*
Strongly typed programming language	2	2	2	6

Author: * *Not integrated in the language, requires external tools/methods (there's a plethora of approaches in the literature)*
** *Have to be stated by the project*

J.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	3	3	3	9
Automatic translation	3	3	3	9
Code Generation	2*	*	0	2*
Model verification	2	*	2	4*
Test generation	2	0	0	2
Simulation, execution, debugging	3	3	3	9
Formal proof	0	3	2*	5

Author: * *The model is itself executable*

Assessor 2:

* *There are some model-checking tools that can accept SystemC code; for example CBMC in the case of a restricted set of properties (see e.g.* `http://www.informatik.uni-bremen.de/agra/doc/konf/10MEMOCODE-tlmpc.pdf`*). Nonetheless I am not sure that we have the expertise for using these prototypes in the project.*

> *Author: I believe that would have to resort to (model-based) testing here, if we want to verify the whole model. But this will be the case anyway for most approaches as we will not be able to formally verify the entire ETCS system. This is simply infeasible. However, we could try to apply model checking to parts of it. The SystemC-frontend of CBMC is currently not supporting full SystemC (there are some restrictions) but the authors are working on it. This is a work done in the context of the Artemis Project* VeTeSS *(*`http://vetess.eu/`*).*

Modelling support

Does the tool provide a textual or a graphical editor ?

It is a textual language. We are investigating in a SysML/UML integration, see above.

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

The model is itself executable with an integrated simulation environment, but there is a variety of tool providers for software synthesis (see above)

Model verification

Which verification on models are provided by the tool?

Simulation, Testing

Test generation

Does the tool allow to generate tests ? For which purpose ?

There are extensions that support generating random tests with constraints.

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ?

Simulation is supported, debugging can be done by any C++ debugger.

Formal proof

Does the tool allow formal proof ? How ?

No, only by means of external tools

J.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open Source (D2.6-X-36)	3	3	3*	9
Portability to operating systems (D2.6-X-37)	3	3	3	9
Cooperation of tools (D2.6-X-38)	*	2♣	2	4*
Robustness (D2.6-X-41)	3	3	3	9
Modularity (D2.6-X-41.1)	3	3	3	9
Documentation management (D.2.6-X-41.2)	**	2	2	4*
Distributed software development (D.2.6-X-41.3)	3***	3	3	9
Simultaneous multi-users (D.2.6-X-41.4)	3***	3	3	9
Issue tracking (D.2.6-X-41.5)	**	1	2	3*
Differences between models (D.2.6-X-41.6)	3***	3	2	8
Version management (D.2.6-X-41.7)	3***	2	2	7
Concurrent version development (D.2.6-X-41.8)	3***	2	2	7
Model-based version control (D.2.6-X-41.9)	****	0	0	0*
Role traceability (D.2.6-X-41.10)	*	0	0	0*
Safety version traceability (D.2.6-X-41.11)	*	0	0	0*
Model traceability (D.2.6-01-035)	*****		0	0*
Tool chain integration	2******	3	3	8
Scalability	3	3	3	9

*Author: * Unclear to me*
*** Not directly; by means of external tools such as Doxygen (or in the case of issue tracking, e.g., GitHub)*
**** By means of versioning systems such as Git or SVN*
***** For SystemC text-based version control is equivalent to model-based version control.*
****** Can possibly be covered by an associated SysML model. In addition, standardised, machine readable comments in the code could be used.*
******* Tool chain integration can be achieved at different levels. E.g., SystemC can*

be the target language from graphical, higher-level languages (e.g., SysML). SystemC models are executable and thus code generation is possibly no issue if we want to obtain just an executable model but no real code running on the target platform (which is out of scope for openETCS).

Assessor 1:

♣ *SystemC may be used in a tool chain cf. hardware synthesis*

Assessor 2:

∗ *Access to SystemC is restricted and subject to the terms of the SystemC Open Source License (`http://www.accellera.org/about/policies/SystemC_Open_Source_License_v3.3.pdf`). YAOSLA = Yet Another Open Source Licence Agreement !*

J.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

Author: We do not have the information to fill this completely. But the broad industrial usage suggests that certifiability should not be an issue.

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	*	3	3	6*
Proof of correctness (D.2.6-01-42.03)	*	0	0	0*
Existing industrial usage	3	3	3	9
Model verification	*	0	0	0*
Test generation	*	0	0	0*
Simulation, execution, debugging	3	3	3	9
Formal proof	*	0	0	0*

Other elements for tool certification

J.11 Other comments

Please to give free comments on the approach.

J.11.1 Assessor 1 summary

SystemC is a powerful tool for modeling concurrent systems. The key feature is that the model is executable with an integrate simulation environment. Moreover, the user just need a C++ compiler and an editor to edit and run the model. The simulation is event-driven with notion of time, so the model may be used for time-simulation. One big advantage of this approach is to be able to use different c++ library like arithmetic library. It is also possible to do architectural exploration. This is not in the scope of the project but this can be useful to speed up some computation by hardware co-processor. SystemC may simulate hardware and software components together.

Furthermore, it is possible to replace the simulation kernel by an appropriate run-time system to then be able to take into account the OBU hardware and to be conform with some OS API.

The integration with the test from SUBSET076 seems quite straightforward and can be easily realized.

The approach proposed with SysML seems promising and shows that SystemC is easily integrated in a tool chain and be good candidate for target code. Nevertheless the translation or the result of the translation should be proven. Moreover, for the moment the description is more at high level and the software should be refined to be embedded. The definition of the refinement is not yet well-defined.

J.11.2 Assessor 2 summary

While SystemC has strong similarities with VHDL and is often only presented as an hardware description language, it is also widely used in the automotive and avionic industries to provide a high-level, executable models for the dynamic architecture of systems. In this respect, its use in the industry is close in spirit to technologies like Matlab/Simulink or even SCADE.

Actually the comparison with Matlab/Simulink is quite interesting since both technologies face similar problems, like for instance a lack of a proper, unambiguous semantics (partly because of the ease with which the language can be extended). They also share similar advantages: very efficient tooling; a large community of users; a very versatile and expressive language; etc.

SystemC can be used for hardware synthesis (e.g. FPGA) or for generating executable code (software); one of its distinctive advantage with respect to the other modeling languages presented in this document is the large collection of simulators that are available. This makes it a good candidate for testing at the subsystem level and for early architecture exploration; e.g. for architecture dimensioning. Nonetheless it is not clear that different simulators will produce equivalent, reproducible results. This is my motivation for attributing a “1” for its applicability during the “Sub-system formal design” phase (because of the *formal*).

In the proposal from TWT and Uni. Rostock, SystemC is used as the target language in a transformation from SysML. When restricted to this particular usage, some of the problems with the approach could be lessened. The problem of proving that this transformation preserves the semantics of SysML still remains. Still, SystemC could be an interesting choice, in parallel with other modeling languages, as a way to provide “technology/design diversity” and therefore have a more dependable system.

Appendix K: Why3

Author Author of the approaches description: David Mentré (Mitsubishi Electric R&D Centre Europe)

K.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name: Why3

Web site: `http://why3.lri.fr/`

Licence: GNU LGPL

Abstract

Short abstract on the approach and tool (10 lines max)

Why3 is a platform for deductive program verification. It provides a rich language for specification and programming, called WhyML, and relies on external theorem provers, both automated and interactive, to discharge verification conditions. Why3 comes with a standard library of logical theories (integer and real arithmetic, Boolean operations, sets and maps, etc.) and basic programming data structures (arrays, queues, hash tables, etc.). A user can write WhyML programs directly and get correct-by-construction OCaml programs through an automated extraction mechanism. WhyML is also used as an intermediate language for the verification of C, Java, or Ada programs.

Publications

Short list of publications on the approach (5 max)

- Why3: Shepherd Your Herd of Provers (BOOGIE 2011) `http://proval.lri.fr/submissions/boogie11.pdf`
- Verifying Two Lines of C with Why3: an Exercise in Program Verification (VSTTE 2012) `http://why3.lri.fr/queens/queens.pdf`
- Why3 – Where Programs Meet Provers (ESOP 2013) `http://hal.inria.fr/hal-00789533`

K.2 Evaluation

The evaluation of this approach has been stop by the author before the end of the benchmark activity:

Author:

Main reason to stop Why3 model: Why3 and GNATprove have roughly the same capabilities but GNATprove is less error prone.

Both GNATprove and Why3 are using a contract approach, with pre and post-conditions on functions (and some kind of data invariant). Both have same expression capabilities with complicated data structures: sum types (record with discriminant in Ada), array, record, etc. GNATprove code can be compiled with an Ada compiler. Why3 can be translated to OCaml (at least for the latest git version). Both are using several automated SMT solvers, thus they are rather easy to use (once correct invariant are written ;-)).

However GNATprove or at least Ada supports tasking, thus making concurrent models can be considered. This is not possible with Why3.

Moreover, Why3 heavily relies on axiomatizations that are fragile: it is easy to make a mistake when writing axioms. GNATprove has a fixed set of expression capabilities within its logical framework, but at least it is much less risky for a regular user which is not expert in formal methods.

Appendix L: GNATprove

Author Author of the approaches description David Mentré (Mitsubishi Electric R&D Centre Europe)

Assessor 1 First assessor of the approaches %%Name - Company%%

Assessor 2 Second assessor of the approaches Matthias Güdemann (Systerel)

In the sequel, main text is under the responsibilities of the author.

Author: Author can add comments using this format at any place.

Assessor 1: First assessor can add comments using this format at any place.

Assessor 2: Second assessor can add comments using this format at any place.

When a note is required, please follow this list :

0 not recommended, not adapted, rejected

1 weakly recommended, adapted after major improvements, weakly rejected

2 recommended, adapted (with light improvements if necessary) weakly accepted

3 highly recommended, well adapted,strongly accepted

***** difficult to evaluate with a note (please add a comment under the table)

All the notes can be commented under each table.

L.1 Presentation

This section gives a quick presentation of the approach and the tool.

Name Ada 2012 language with GNATprove tool

Web site `http://www.open-do.org/projects/hi-lite/gnatprove/`

Licence GNU GPL

Abstract

Part of the Hi-Lite project, GNATprove is a formal verification tool for Ada, based on the GNAT compiler. It can prove that subprograms respect their contracts, expressed as preconditions and postconditions in the syntax of Ada 2012. The tool automatically discovers the subset of subprograms which can be formally analyzed. GNATprove is currently available for x86 linux, x86 windows and x86-64 linux.

The GNATprove tool can be combined with regular testing tools to cover the whole program using the most efficient approaches.

Publications

- Hi-Lite: The Convergence of Compiler Technology and Program Verification `http://www.open-do.org/wp-content/uploads/2012/11/HILT_2012.pdf`
- Integrating Formal Program Verication with `Testinghttp://www.open-do.org/wp-content/uploads/2011/12/hi-lite-erts2012.pdf`

L.2 Main usage of the approach

This section discusses the main usage of the approach.

According to the figure 1, for which phases do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
System Analysis	1		1	
Sub-system formal design	2		1	
Software design	3		3	
Software code generation	3		3	

According to the figure 1, for which type of activities do you recommend the approach (give a note from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Documentation	1		1	
Modeling	1		1	
Design	2		1	
Code generation	3		3	
Verification	3		3	
Validation	3		3	
Safety analyses	1		1	

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives?

Ada (and its SPARK variant) are used for avionics DO-178B projects, EN50128 railway projects (Alstom, Siemens, ...), ...

L.3 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	Author	Assessor 1	Assessor 2	Total
Informal language	0		0	
Semi-formal language	0		0	
Formal language	3		3	
Structured language	3		3	
Modular language	3		3	
Textual language	3		3	
Mathematical symbols or code	3		3	
Graphical language	0		0	

According WP2 requirements, give a note for the capabilities of the language (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Declarative formalization of properties (D.2.6-X-28)	3		3	
Simple formalization of properties (D.2.6-X-28.1)	2		2	
Scalability : capability to design large model	3		3	
Easily translatable to other languages (D.2.6-X-30)	2		2	
Executable directly (D.2.6-X-33)	3		3	
Executable after translation to a code (D.2.6-X-33) (precise if the translation is automatic)	3 3[1]		3*	
Simulation, animation (D.2.6-X-33)	1		2*	
Easily understandable (D.2.6-X-27)	2		2	
Expertise level needed (0 High level, 3 few level)	2		2	
Standardization (D.2.6-X-29)	3		3	
Documented (D.2.6-X-29)	3		3	
Extensible language (D.2.6-01-28)	3		3	

Assessor 2 Executable after translation to a code is not applicable, as it is directly executable code, i.e., no intermediate step is needed for compilation.

Simulation, animation should be possible with the GNAT Visual Debugger.

Documentation

Describe how the language is documented, the existing guidelines, coding rules, standardization...

Ada 2012 is described in ISO/IEC 8652:2012 standard. It was developed under USA's Department of Defense. Several documents describe coding guidelines.

Language usage

Describe the possible restriction on the language

The Ada language is a general purpose programming language.

Assessor 2 For GNATprove / SPARK 2014, only a subset of full Ada can be used, nevertheless this still represents a general purpose programming language.

L.4 System Analysis

This section discusses the usage of the approach for system analysis. It can be skipped depending the results of L.8.

According to WP2 requirements, how the approach can be involved for the sub-system requirement specification?

	Author	Assessor 1	Assessor 2	Total
Independent System functions definition (D.2.6-X-10.2.1)	1		1	
System architecture design (D.2.6-X-10.2)	2		2	
System data flow identification (D.2.6-X-10.2.3)	2		2	
Sub-system focus (D.2.6-X-10.2.4)	1		1	
System interfaces definition (D.2.6-X-10.2.5)	2		2	
System requirement allocation (D.2.6-X-10.3)	0		0	
Traceability with SRS (D.2.6-X-10.5)	1		1	
Traceability with Safety activities (D.2.6-X-11)	2		2	

L.5 Sub-System formal design

This section discusses the usage of the approach for sub-system formal design. It can be skipped depending the results of L.8.

Two kinds of model can be planned during this phase: semi-formal models to cover the SSRS (D.2.6-X-12.1) and strictly formal models to focuss on some functional and safety aspects (D.2.6-X-14). Obviously some strictly formal means can be used to define the semi-formal model.

L.5.1 Semi-formal model

Concerning semi-formal model, how the WP2 requirements are covered?

	Author	Assessor 1	Assessor 2	Total
Consistency to SSRS (D.2.6-X-12.2)	2		2	
Coverage of SSRS (D.2.6-X-12.2.1)	3		2	
Coverage of SSHA (D.2.6-X-12.2.2)	2		2	
Management of requirement justification (D.2.6-X-12.2.3)	0		0	
Traceability to SSRS (D.2.6-X-12.2.5)	1		1	
Traceability of exported requirements (D.2.6-X-12.2.6)	1		1	
Simulation or animation (D.2.6-X-13 partial)	3		3	
Execution (D.2.6-X-13 partial)	3		3	
Extensible to strictly formal model (D.2.6-X-14.3)	3		3	
Easy to refine towards strictly formal model (D.2.6-X-14.4)	3		3	
Extensible and modular design (D.2.6-X-15)	3		3	
Extensible to software architecture and design (D.2.6-X-30)	3		3	

Concerning safety properties management, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Safety function isolation (D.2.6-X-17)	2		2	
Safety properties formalisation (D.2.6-X-22)	3		2*	
Logical expression (D.2.6-X-28.2.2)	3		3	
Timing constraints (D.2.6-X-28.2.3)	1		1	
Safety properties validation (D.2.6-X-23.2)	3		2*	
Logical properties assertion (D.2.6-X-34)	3		3	
Check of assertions (D.2.6-X-34.1)	3		3	

Assessor 2 Safety properties formalization and validation is probably not easy if it cannot be reduced to pre-/ postconditions of a single function. If several functions are concerned, I am not sure how it can be assured that the conjunction of the post-conditions correctly implies a higher level safety property.

Does the language allow to formalize (D.2.6-X-31):

	Author	Assessor 1	Assessor 2	Total
State machines	3		3	
Time-outs	1		1*	
Truth tables	3		3	
Arithmetic	3		3	
Braking curves	3		3	
Logical statements	3		3	
Message and fields	3		3	

Assessor 2 Ada offers time-outs, is this not within the subset supported by SPARK 2014?

Additional comments on semi-formal model

Do you think your semi-formal model is sufficient to cover a safe design of the on-board unit until code generation ? All comments on links to other models, validation and verification activities are welcomed.

It is possible to make a semi-formal model using Ada 2012 but this approach is a bit far fetched for the approach.

L.5.2 Strictly formal model

Concerning strictly formal model, how the WP2 requirements are covered ?

	Author	Assessor 1	Assessor 2	Total
Consistency to SFM (D.2.6-X-14.2)	2		2	
Coverage of SSRS (D.2.6-X-14.2)	3		3	
Traceability to SSRS (D.2.6-X-14.3)	1		1	
Extensible to software design (D.2.6-X-16)	3		3	
Safety function isolation (D.2.6-X-17)	3		2	
Safety properties formalisation (D.2.6-X-22)	3		2	
Logical expression (D.2.6-X-28.2.2)	3		3	
Timing constraints (D.2.6-X-28.2.3)	1		1	
Safety properties validation (D.2.6-X-23.3)	3		3	
Logical properties assertion (D.2.6-X-34)	3		3	
Proof of assertions (D.2.6-X-34.2)	3		3	

Does the language allow to formalize (D.2.6-X-32):

	Author	Assessor 1	Assessor 2	Total
State machines	3		3	
Time-outs	1		1	
Truth tables	3		3	
Arithmetic	3		3	
Braking curves	3		3	
Logical statements	3		3	
Message and fields	3		3	

Additional comments on semi-formal model

Do you think your strictly formal model can be directly defined from the SSRS ? All comments on links to other models, validation and verification activities are welcomed.

Ada language is suitable to formalize a strictly formal model. As Ada is a programming language, a design phase should probably be put between the SSRS and the strictly formal model.

L.6 Software design

This section discusses the usage of the approach for software design. It can be skipped depending the results of L.8.

L.6.1 Functional design

How the approach allows to produce a functional software model of the on-board unit ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal model	3		3	
Software architecture description	3		2	
Software constraints	3		3	
Traceability	1		1	
Executable	3		3	

L.6.2 SSIL4 design

How the approach allows to produce in safety a software model ?

	Author	Assessor 1	Assessor 2	Total
Derivation from system semi-formal or strictly formal model	3		3	
Software architecture description	3		3	
Software constraints	3		3	
Traceability	1		1	
Executable	3		3	
Conformance to EN50128 § 7.2	1		1	
Conformance to EN50128 § 7.3	3		2	
Conformance to EN50128 § 7.4	3		3	

Which criteria for software architecture are covered by the methodology (see EN50128 table A.3) :

	Author	Assessor 1	Assessor 2	Total
Defensive programming	3		3	
Fault detection & diagnostic	3		3	
Error detecting code	3		3	
Failure assertion programming	3		3	
Diverse programming	3		3	
Memorising executed cases	3		3	
Software error effect analysis	1		1	
Fully defined interface	3		3	
Modelling	2		2	
Structured methodology	1		1	

L.7 Software code generation

This section discusses the usage of the approach for software code generation. It can be skipped depending the results of L.8.

Which criteria for software design and implementation are covered by the methodology (see EN50128 table A.4) :

	Author	Assessor 1	Assessor 2	Total
Formal methods	3		3	
Modeling	1		1	
Modular approach (mandatory)	3		3	
Components	3		3	
Design and coding standards (mandatory)	3		3	
Strongly typed programming language	3		3	

L.8 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	Author	Assessor 1	Assessor 2	Total
Modelling support	1		1	
Automatic translation	-		*	
Code Generation	3		3	
Model verification	3		3	
Test generation	2		2	
Simulation, execution, debugging	3		3	
Formal proof	3		3*	

Assessor 2 The formal proofs are made by using external tools.

Modelling support

Does the tool provide a textual or a graphical editor ?

Textual editor.

Automatic translation and code generation

Which translation or code generation is supported by the tool ?

Ada code is directly compilable to machine assembly.

Model verification

Which verification on models are provided by the tool?

Any contract expressed in first order logic can be verified by the tool.

Test generation

Does the tool allow to generate tests ? For which purpose ?

The tool allows to generate test benches. Tests themselves should be generated manually.

Simulation, execution, debugging

Does the tool allow to simulate or to debbug step by step a model or a code ?

Yes, tool as debugging facilities.

Formal proof

Does the tool allow formal proof ? How ?

Yes. Properties to verify are expressed as first order contracts and code annotations (loops, assertions). An automatic prover can then be used to prove those contracts and annotations.

Assessor 2 How well are interactive proofs supported?

L.9 Use of the tool

According WP2 requirements, give a note for characteristics of the use of the tool (from 0 to 3) :

	Author	Assessor 1	Assessor 2	Total
Open SourceAssessor 2	Total			
(D2.6-X-36)	3		2*	
Portability to operating systems (D2.6-X-37)	3		3	
Cooperation of tools (D2.6-X-38)	3		3	
Robustness (D2.6-X-41)	3		3	
Modularity (D2.6-X-41.1)	3		3	
Documentation management (D.2.6-X-41.2)	3		3	
Distributed software development (D.2.6-X-41.3)	3		3	
Simultaneous multi-users (D.2.6-X-41.4)	3		3	
Issue tracking (D.2.6-X-41.5)	0		0	
Differences between models (D.2.6-X-41.6)	1*		2	
Version management (D.2.6-X-41.7)	0		1*	
Concurrent version development (D.2.6-X-41.8)	3		3	
Model-based version control (D.2.6-X-41.9)	-		*	
Role traceability (D.2.6-X-41.10)	-			
Safety version traceability (D.2.6-X-41.11)	3		3	
Model traceability (D.2.6-01-035)	1		1	
Tool chain integration	2		2	
Scalability	3		3	

Assessor 2 The tool is free only for non-commercial or free software development.

The development tool GPS supports a visual file diff view and different version management systems.

L.10 Certifiability

This section discusses how the tool can be classified according EN50128 requirements (D.2.6-X-50).

	Author	Assessor 1	Assessor 2	Total
Tool manual (D.2.6-01-42.02)	3		3	
Proof of correctness (D.2.6-01-42.03)	?		*	
Existing industrial usage	3		3	
Model verification	3		3	
Test generation	1		*	
Simulation, execution, debugging	2		2	
Formal proof	3		3	

Assessor 2 I did not look into test generation for GNATprove / SPARK 2013 with the GPS tool.

Other elements for tool certification

The Ada tool chain is used for certification of aeronautics D0178B products and railway EN50128 products.

L.11 Other comments

Please to give free comments on the approach.