OETCS/WP7/O7.1.3_O7.1.7 – 00/02

openETCS

ITEA2 Project
Call 6 11025
2012 – 2015

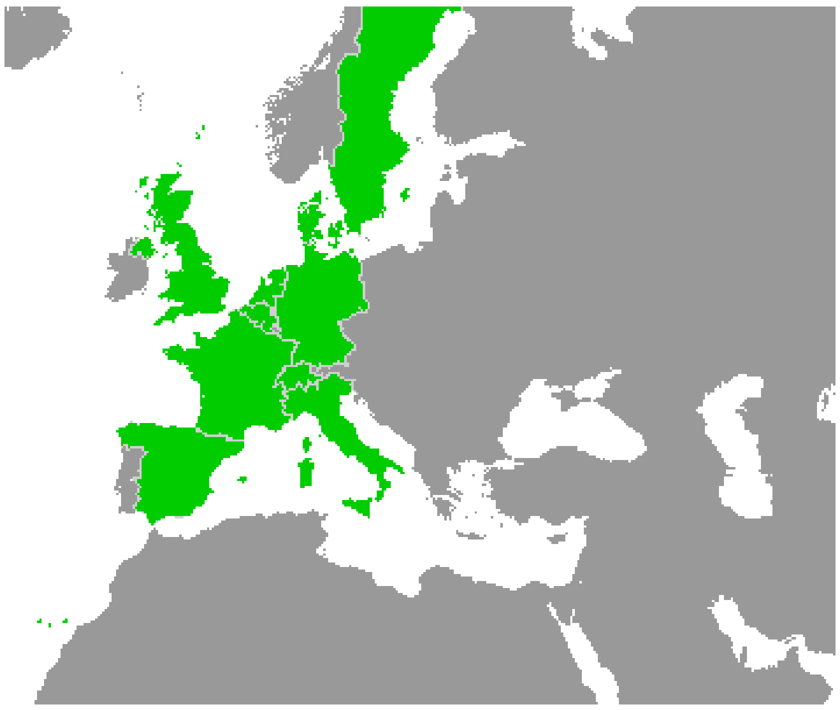
Work-Package 7: “Primary tool chain”

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



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

Figures

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 :

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

For each approach and tool, the initial author of the evaluation is the partner in charge of the modelling. Two assessors, for each approaches, are in charge of the review of the evaluation and can correct it or add comments.

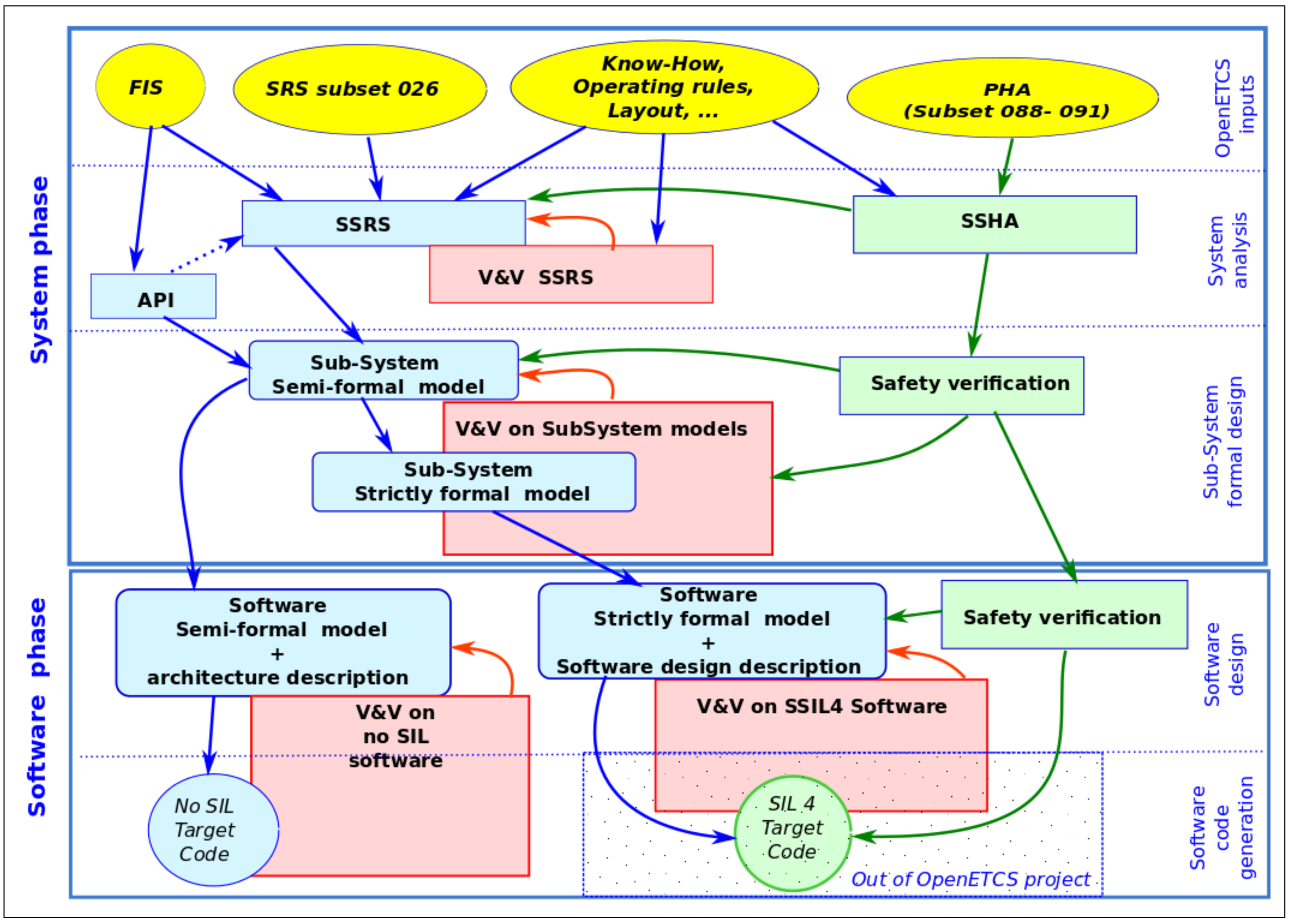


Figure 1. Main OpenETCS process

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.

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) :

	CORE	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
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) :

	CORE	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Documentation											
Modeling											
Design											
Code generation											
Verification											
Validation											
Safety analyses											

3.2 Language

This section discusses the main element of the language.

Which are the main characteristics of the language :

	CORE	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
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) :

	CORE	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)											
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)											

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 ?

	CORE	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)											
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)											

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 ?

	CORE	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)											
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-15)											

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

	CORE	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)											
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):

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

3.4.2 Strictly formal model

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

	CORE	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)											
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):

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

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 ?

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

3.5.2 SSIL4 design

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

	CORE	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											
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) :

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

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) :

	CORE	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Formal methods											
Modeling											
Modular approach (mandatory)											
Components											
Design and coding standards (mandatory)											
Strongly typed programming language											

3.7 Main usage of the tool

This section discusses the main usage of the tool.

Which task are covered by the tool ?

	CORE	GOPRR	ERTMSFormalSpecs	SysML with Papyrus	SysML with Entreprise Architect	SCADE	EventB	Classical B	Petri Nets	System C	GNATprove
Modelling support											
Automatic translation											
Code Generation											
Model verification											
Test generation											
Simulation, execution, debugging											
Formal proof											

3.8 Use of the tool

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

3.9 Certifiability

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

	CORE	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)											
Proof of correctness (D.2.6-01-42.03)											
Existing industrial usage											
Model verification											
Test generation											
Simulation, execution, debugging											
Formal proof											

Appendix A: CORE Workstation 5.1

Author Author of the approaches description: Cyril Cornu (All4Tec)

Assessor 1 First assessor of the approaches Marielle Petit-Doche (Systerel)

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.

A.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)

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				
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 ?

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				
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

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)				
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)				

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

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.

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)				
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.

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				
Software constraints				
Traceability				
Executable				

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				
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				

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				
Modeling				
Modular approach (mandatory)				
Components				
Design and coding standards (mandatory)				
Strongly typed programming language				

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				
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 ?

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)				
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				

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)				
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

A.11 Other comments

Please to give free comments on the approach.

Appendix B: GOPRR

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

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

Assessor 2 Second assessor of the approaches Matthias Gudemann (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.

B.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)

B.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 ?

B.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

B.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)				

B.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.

B.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.

B.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.

B.6 Software design

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

B.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				

B.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				

B.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				

B.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 ?

B.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				

B.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

B.11 Other comments

Please to give free comments on the approach.

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/

Licence 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.

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			
Sub-system formal design	3			
Software design	0			
Software code generation	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.

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			
Modeling	3			
Design	3			
Code generation	0			
Verification	3			
Validation	3			
Safety analyses	0			

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			
Semi-formal language	3			
Formal language	3			
Structured language	3			
Modular language	3			
Textual language	3			
Mathematical symbols or code	3			
Graphical language	3			

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			
Simple formalization of properties (D.2.6-X-28.1)	2			
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)	3			
Executable after translation to a code (D.2.6-X-33) (precise if the translation is automatic)	3 2			
Simulation, animation (D.2.6-X-33)	3			
Easily understandable (D.2.6-X-27)	3			
Expertise level needed (0 High level, 3 few level)	2			
Standardization (D.2.6-X-29)	3			
Documented (D.2.6-X-29)	3			
Extensible language (D.2.6-01-28)	3			

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			
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)	3			
Traceability with SRS (D.2.6-X-10.5)	3			
Traceability with Safety activities (D.2.6-X-11)	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.

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 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.

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			
Coverage of SSRS (D.2.6-X-12.2.1)	3			
Coverage of SSHA (D.2.6-X-12.2.2)	3			
Management of requirement justification (D.2.6-X-12.2.3)	3			
Traceability to SSRS (D.2.6-X-12.2.5)	3			
Traceability of exported requirements (D.2.6-X-12.2.6)	3			
Simulation or animation (D.2.6-X-13 partial)	3			
Execution (D.2.6-X-13 partial)	3			
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			

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			
Safety properties formalisation (D.2.6-X-22)	2			
Logical expression (D.2.6-X-28.2.2)	2			
Timing constraints (D.2.6-X-28.2.3)	2			
Safety properties validation (D.2.6-X-23.2)	2			
Logical properties assertion (D.2.6-X-34)	2			
Check of assertions (D.2.6-X-34.1)	2			

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

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

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

	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.

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 § 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				

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.

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			
Automatic translation	1			
Code Generation	1			
Model verification	3			
Test generation	1			
Simulation, execution, debugging	3			
Formal proof	0			

Modelling support

Does the tool provide a textual or a graphical editor ?

Author. Both.

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 debbug 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			
Portability to operating systems (D2.6-X-37)	2			
Cooperation of tools (D2.6-X-38)	3			
Robustness (D2.6-X-41)	3			
Modularity (D2.6-X-41.1)	3			
Documentation management (D.2.6-X-41.2)	2			
Distributed software development (D.2.6-X-41.3)	2			
Simultaneous multi-users (D.2.6-X-41.4)	1			
Issue tracking (D.2.6-X-41.5)	3			
Differences between models (D.2.6-X-41.6)	1			
Version management (D.2.6-X-41.7)	2			
Concurrent version development (D.2.6-X-41.8)	2			
Model-based version control (D.2.6-X-41.9)	2			
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)	3			
Tool chain integration	2			
Scalability	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			
Proof of correctness (D.2.6-01-42.03)	2			
Existing industrial usage	0			
Model verification	3			
Test generation	0			
Simulation, execution, debugging	3			
Formal proof	0			

Other elements for tool certification

C.11 Other comments

Please to give free comments on the approach.

Appendix D: SysML with Papyrus

Author Author of the approaches description Alexander Stante (Fraunhofer)

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

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, 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 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)

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				
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 ?

D.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

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-10.2.1)				
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)				

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 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.

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)				
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.

D.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.

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				
Software architecture description				
Software constraints				
Traceability				
Executable				

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				
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				

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				
Modeling				
Modular approach (mandatory)				
Components				
Design and coding standards (mandatory)				
Strongly typed programming language				

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				
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 ?

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)				
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				

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)				
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

D.11 Other comments

Please to give free comments on the approach.

Appendix E: SysML with Entreprise 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 modeiling 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			
Sub-system formal design	3			
Software design	3			
Software code generation	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.

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			
Modeling	3			
Design	3			
Code generation	2			
Verification	1			
Validation	1			
Safety analyses	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.

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			
Semi-formal language	2			
Formal language	3			
Structured language	3			
Modular language	3			
Textual language	1			
Mathematical symbols or code	1			
Graphical language	3			

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			
Simple formalization of properties (D.2.6-X-28.1)	2			
Scalability : capability to design large model	3			
Easily translatable to other languages (D.2.6-X-30)	2			
Executable directly (D.2.6-X-33)	1			
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)	1			
Easily understandable (D.2.6-X-27)	3			
Expertise level needed (0 High level, 3 few level)	2			
Standardization (D.2.6-X-29)	3			
Documented (D.2.6-X-29)	3			
Extensible language (D.2.6-01-28)	3			

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			
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)	3			
Traceability with SRS (D.2.6-X-10.5)	3			
Traceability with Safety activities (D.2.6-X-11)	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			
Coverage of SSRS (D.2.6-X-12.2.1)	3			
Coverage of SSHA (D.2.6-X-12.2.2)	*			
Management of requirement justification (D.2.6-X-12.2.3)	3			
Traceability to SSRS (D.2.6-X-12.2.5)	3			
Traceability of exported requirements (D.2.6-X-12.2.6)	3			
Simulation or animation (D.2.6-X-13 partial)	2			
Execution (D.2.6-X-13 partial)	2			
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. 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			
Safety properties formalisation (D.2.6-X-22)	2			
Logical expression (D.2.6-X-28.2.2)	2			
Timing constraints (D.2.6-X-28.2.3)	3			
Safety properties validation (D.2.6-X-23.2)	1			
Logical properties assertion (D.2.6-X-34)	2			
Check of assertions (D.2.6-X-34.1)	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			
Time-outs	3			
Truth tables	1			
Arithmetic	2			
Braking curves	2			
Logical statements	3			
Message and fields	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			
Coverage of SSRS (D.2.6-X-14.2)		3		
Traceability to SSRS (D.2.6-X-14.3)	3			
Extensible to software design (D.2.6-X-16)	3			
Safety function isolation (D.2.6-X-17)	*			
Safety properties formalization (D.2.6-X-22)	3*			
Logical expression (D.2.6-X-28.2.2)	2			
Timing constraints (D.2.6-X-28.2.3)	3			
Safety properties validation (D.2.6-X-23.3)	1			
Logical properties assertion (D.2.6-X-34)	2			
Proof of assertions (D.2.6-X-34.2)	1			

Author.

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

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

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

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			
Software architecture description	3			
Software constraints	3			
Traceability	3			
Executable	2			

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			
Software architecture description	3			
Software constraints	3			
Traceability	3			
Executable	2			
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	*			
Fault detection & diagnostic	*			
Error detecting code	*			
Failure assertion programming	3			
Diverse programming	3			
Memorising executed cases	*			
Software error effect analysis	*			
Fully defined interface	3			
Modelling	3			
Structured methodology	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 ...

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	*			
Modeling	3			
Modular approach (mandatory)	3			
Components	3			
Design and coding standards (mandatory)	3			
Strongly typed programming language	3			

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			
Automatic translation	3			
Code Generation	2			
Model verification	1			
Test generation	1			
Simulation, execution, debugging	1			
Formal proof	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			
Portability to operating systems (D2.6-X-37)	2			
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)	2			
Distributed software development (D.2.6-X-41.3)	3			
Simultaneous multi-users (D.2.6-X-41.4)	1			
Issue tracking (D.2.6-X-41.5)	2			
Differences between models (D.2.6-X-41.6)	1			
Version management (D.2.6-X-41.7)	3			
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)	3			
Safety version traceability (D.2.6-X-41.11)	0			
Model traceability (D.2.6-01-035)	3			
Tool chain integration	2*			
Scalability	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.

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

Other elements for tool certification

E.11 Other comments

Please to give free comments on the approach.

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			
Sub-system formal design	3			
Software design	3			
Software code generation	3			

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			
Modeling	3			
Design	3			
Code generation	3			
Verification	3			
Validation	3			
Safety analyses	0			

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			
Semi-formal language	0			
Formal language	3			
Structured language	3			
Modular language	3			
Textual language	3			
Mathematical symbols or code	3			
Graphical language	3			

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

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.*

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			
System architecture design (D.2.6-X-10.1.2)	1			
System data flow identification (D.2.6-X-10.2.3)	3			
Sub-system focus (D.2.6-X-10.2.4)	2			
System interfaces definition (D.2.6-X-10.2.5)	2			
System requirement allocation (D.2.6-X-10.3)	3			
Traceability with SRS (D.2.6-X-10.5)	3			
Traceability with Safety activities (D.2.6-X-11)	3			

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			
Coverage of SSRS (D.2.6-X-12.2.1)	3			
Coverage of SSHA (D.2.6-X-12.2.2)	3			
Management of requirement justification (D.2.6-X-12.2.3)	3			
Traceability to SSRS (D.2.6-X-12.2.5)	3			
Traceability of exported requirements (D.2.6-X-12.2.6)	3			
Simulation or animation (D.2.6-X-13 partial)	3			
Execution (D.2.6-X-13 partial)	3			
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			

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			
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-23.2)	2			
Logical properties assertion (D.2.6-X-34)	2			
Check of assertions (D.2.6-X-34.1)	2			

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

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

	Author	Assessor 1	Assessor 2	Total
State machines	3			
Time-outs	3			
Truth tables	3			
Arithmetic	3			
Braking curves	3			
Logical statements	3			
Message and fields	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			
Coverage of SSRS (D.2.6-X-14.2)	3			
Traceability to SSRS (D.2.6-X-14.3)	3			
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)	3			
Safety properties validation (D.2.6-X-23.3)	2			
Logical properties assertion (D.2.6-X-34)	2			
Proof of assertions (D.2.6-X-34.2)	2			

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

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

Author: SCADE is targeted for these purposes.

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			
Software architecture description	3			
Software constraints	3			
Traceability	3			
Executable	3			

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			
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	0			
Error detecting code	0			
Failure assertion programming	1			
Diverse programming	0			
Memorising executed cases	0			
Software error effect analysis	0			
Fully defined interface	3			
Modelling	3			
Structured methodology	3			

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			
Modeling	3			
Modular approach (mandatory)	3			
Components	3			
Design and coding standards (mandatory)	3			
Strongly typed programming language	3			

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			
Automatic translation	3			
Code Generation	3			
Model verification	3			
Test generation	2			
Simulation, execution, debugging	3			
Formal proof	3			

Modelling support

Does the tool provide a textual or a graphical editor ? 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, test 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 observer configuration.

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			
Portability to operating systems (D2.6-X-37)	3			
Cooperation of tools (D2.6-X-38)	3			
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)	1			
Issue tracking (D.2.6-X-41.5)	0			
Differences between models (D.2.6-X-41.6)	3			
Version management (D.2.6-X-41.7)	3			
Concurrent version development (D.2.6-X-41.8)	2			
Model-based version control (D.2.6-X-41.9)	3			
Role traceability (D.2.6-X-41.10)	0			
Safety version traceability (D.2.6-X-41.11)	0			
Model traceability (D.2.6-01-035)	3			
Tool chain integration	3			
Scalability	3			

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			
Proof of correctness (D.2.6-01-42.03)	3			
Existing industrial usage	3			
Model verification	3			
Test generation	2			
Simulation, execution, debugging	3			
Formal proof	3			

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 (Systerel)

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

Assessor 2 Second assessor of the approaches Stanislas Pinte (ERTMS Solutions)

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 %%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)

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				
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 ?

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				
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

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)				
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)				

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

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.

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)				
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.

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.

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.

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				
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 ?

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)				
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				

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)				
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

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 railroad industry, to develop complex systems. It covers software development from formal specifications 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 contraints.

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	0			
Sub-system formal design	2			
Software design	3			
Software code generation	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	0			
Modeling	3			
Design	3			
Code generation	3			
Verification	3			
Validation	2			
Safety analyses	1			

Known usages

Have you some examples of usage of this approach to compare with the OpenETCS objectives ? Classical B has been used successfully in railway industry (mainly by Alstom, aiemens 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. 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... Language is documented with language manual reference. Industrial have developed their own coding rules and guidelines.

Language usage

Describe the possible restriction on the language

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			

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

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			

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			

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			

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.

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			

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			

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.

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 ? Textual editor

Automatic translation and code generation

Which translation or code generation is supported by the tool ? 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? simulation, Validation and formal proof, test coverage

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 ? Code can be simulated step by step with specific tool of the target language.

Formal proof

Does the tool allow formal proof ? How ?

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			

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

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 %%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.

I.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)

I.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 ?

I.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

I.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)				

I.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.

I.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.

I.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.

I.6 Software design

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

I.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				

I.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				

I.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				

I.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 ?

I.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				

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)				
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

I.11 Other comments

Please to give free comments on the approach.

Appendix J: System C

This is a preliminary version of the description and there are still some open questions, also due to unclarities in this template.

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 DalZilio / 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 priciples 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			
Sub-system formal design	3			
Software design	3			
Software code generation	2			

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	2			
Validation	2			
Safety analyses	2			

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			
Semi-formal language	3			
Formal language	2			
Structured language	3			
Modular language	3			
Textual language	3			
Mathematical symbols or code	3			
Graphical language	*			

Author. ** Not graphical, but we are investigating SystemC and UML/SysML integration, the ITEA MARTES Project is addressing this aspect*

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)	1			
Simple formalization of properties (D.2.6-X-28.1)	1			
Scalability : capability to design large model	3			
Easily translatable to other languages (D.2.6-X-30)	2			
Executable directly (D.2.6-X-33)	3			
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)	3			

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.

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)	**			
Traceability with Safety activities (D.2.6-X-11)	2*			

Author: * *Can possibly be covered by an associated SysML model. In addition, standardised, machine readable comments in the code could be used.*
*** This is not the scope of SystemC*

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			
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)	2			
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)	3			
Execution (D.2.6-X-13 partial)	3			
Extensible to strictly formal model (D.2.6-X-14.3)	2			
Easy to refine towards strictly formal model (D.2.6-X-14.4)	2			
Extensible and modular design (D.2.6-X-15)	3			
Extensible to software architecture and design (D.2.6-X-30)	3			

*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"?*

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			
Logical expression (D.2.6-X-28.2.2)	3			
Timing constraints (D.2.6-X-28.2.3)	3			
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. * Item not clear to me, should be a requirement for the actual implementation, not a model*

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

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

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.

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.

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	*			
Software architecture description	3			
Software constraints	3			
Traceability	2**			
Executable	3			

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.*

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	*			
Software architecture description	3			
Software constraints	3			
Traceability	2**			
Executable	3			
Conformance to EN50128 § 7.2	***			
Conformance to EN50128 § 7.3	***			
Conformance to EN50128 § 7.4	***			

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?*

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			
Error detecting code	3			
Failure assertion programming	3			
Diverse programming	*			
Memorising executed cases	3			
Software error effect analysis	*			
Fully defined interface	3			
Modelling	3			
Structured methodology	3			

Author. ** SystemC is a language and no methodology. These methodologies can be applied for most languages.*

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*			
Modeling	3			
Modular approach (mandatory)	3			
Components	3			
Design and coding standards (mandatory)	**			
Strongly typed programming language	2			

*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			
Automatic translation	3			
Code Generation	2*			
Model verification	2			
Test generation	2			
Simulation, execution, debugging	3			
Formal proof	0			

*Author. * The model is itself executable*

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

Author. ** Unclear to me*
*** 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).*

J.10 Certifiability

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

Author. I have no clue here.

	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

J.11 Other comments

Please to give free comments on the approach.

Appendix K: UPPAAL

Author Author of the approaches description Stefan Rieger (TWT)

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

Assessor 2 Second assessor of the approaches Silvano DalZilio / 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.

K.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)

K.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 ?

K.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

K.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)				

K.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.

K.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.

K.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.

K.6 Software design

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

K.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				

K.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				

K.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				

K.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 ?

K.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				

K.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

K.11 Other comments

Please to give free comments on the approach.

Appendix L: GNATprove

Author Author of the approaches description David Mentré (MERCE)

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

Assessor 2 Second assessor of the approaches Matthias Gudemann (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 %%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)

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				
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 ?

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				
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

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.

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)				

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)				
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.

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)				
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.

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				
Software architecture description				
Software constraints				
Traceability				
Executable				

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				
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				

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				
Modeling				
Modular approach (mandatory)				
Components				
Design and coding standards (mandatory)				
Strongly typed programming language				

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				
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 ?

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 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				

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)				
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

L.11 Other comments

Please to give free comments on the approach.