OETCS/WP7

openETCS

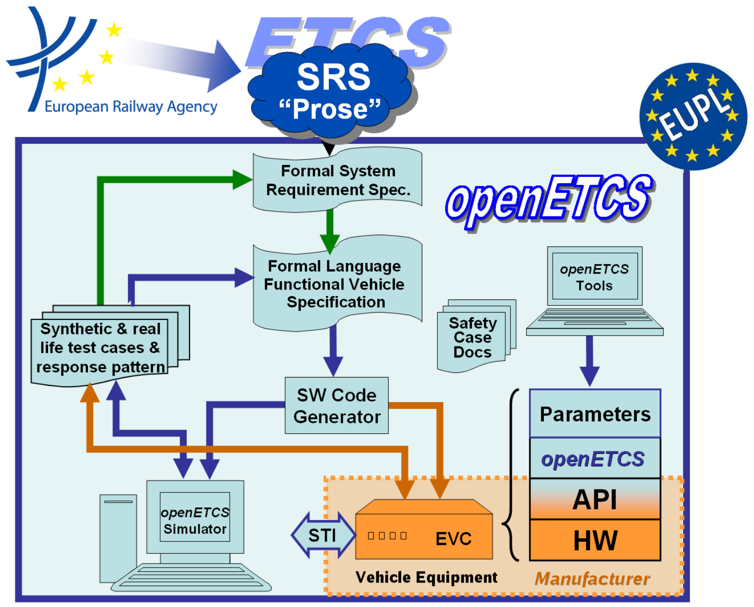
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 7: “Tool chain”

openETCS:
Migration from Scade to an Open Alternative

Silvano Dal Zilio, Marc Behrens, Raphaël Faudou, David Mentré
and Xavier Zeitoun

October 2015



Funded by:

Federal Ministry
of Education
and Research

Région de
Bruxelles-
Capitale

GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, ENERGÍA
Y TURISMO

This page is intentionally left blank

Work-Package 7: “Tool chain”

OETCS/WP7
October 2015

openETCS: Migration from Scade to an Open Alternative

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Silvano Dal Zilio (LAAS-CNRS)	signature ()	signature ()	signature ()

Silvano Dal Zilio

LAAS-CNRS
7 ave. Colonel Roche
F-31004 Toulouse, France
eMail: dalzilio@laas.fr
WebSite: www.laas.fr

Marc Behrens

DLR

Raphaël Faudou

SAMARES Engineering

David Mentré

MERCE

Xavier Zeitoun

CEA

Output Document

Prepared for openETCS@ITEA2 Project

Abstract: This document evaluates possible scenarios for migrating the openETCS OBU model, currently developed using the proprietary Scade language, to an open format. After a brief description of the current modeling choices, we describe the different export formats supported by the Scade toolsuite. We base our possible migration scenarios on an analysis of the existing tools that can exploit these output formats.

Table of Contents

Document Control

Document information	
Work Package Deliverable ID	WP7
Document title Document version Document authors (org.)	openETCS Migration from Scade to an Open Alternative 1.1 Silvano Dal Zilio (LAAS-CNRS)

Review information	
Last version reviewed	01
Main reviewers (org.)	Marc Behrens (DLR); Raphaël Faudou (SAMARES); David Mentré (MERCE); Xavier Zeitoun (CEA)

Approbation			
	Name	Role	Date
Written by			October 2015
Approved by	–	–	

Document evolution			
Version	Date	Author(s)	Justification
01.0	October 2015	S. Dal Zilio	creation
01.1	November 2015	S. Dal Zilio	added comments from reviewers

1 Introduction

One of the main purpose of the openETCS project is to develop a model of the ETCS on-board unit (OBU) using "Open Standards" at every stages. This includes generating open hardware and software specifications, writing interface definition using open languages, and developing open source design tools for supporting this effort. One of the most important output of the project is a formal specification of the ETCS that follows the specification defined by the European Railway Agency (ERA) in its Subset-026 document [1]. Nonetheless, for reasons that we will briefly recall in this chapter, the proprietary language Scade—that is a "closed-source" solution—was chosen to develop the functional model of the OBU.

Scade is a formal modeling language and tool suite targeting safety-critical embedded systems. It is developed by Esterel Technologies, now a company of the ANSYS group (`http://www.ansys.com/`). Scade has been used for more than 15 years to develop a broad range of control applications in the avionics, rail, and automotive domains. The current Scade model for the OBU is a major output of the project. The architectural model (modeled in SysML using the Scade system tool) and the functional model (modeled in Scade using the Scade suite tool) have been developed from scratch—without the use of prior components—and cover all the aspects defined in chapters 3, 4,5, 7 and 8 of the *ETCS specification*[1]. The models are made available on the GitHub of the openETCS project (`https://github.com/openETCS/modeling/tree/master/model`) and a complete description is given in deliverable D3.5.3 [2].

Several natural questions arise if we want to meet the "open philosophy" of the openETCS project. For instance, would it be possible to develop a model equivalent to what has been achieved with Scade using an open source modeling language and open source tooling? More interestingly, is it possible to capitalize on the substantial modeling effort already invested in the Scade model and to migrate this model into an open source format? We concentrate on this last question in this report.

Solving this migration problem raises several issues, for instance concerning the simplicity of the migration process or the coherence of the resulting models. Concerning the first point, *simplicity*, we are of course interested by methods that are fully automatic and, if possible, at least partially available (implemented). This is why we will concentrate on solutions based on a source-level conversion from Scade. Concerning the validity of the (target) models—and this is certainly the critical stumbling block raised in this report—we need to evaluate if all the hypotheses used to check the validity of the model (that is to check how close the model is to the informal specification of Subset-026) are safely "carried-over" during the migration process. In the remainder of this text we use the term of *semantics preserving* migration to refer to this issue. This issue is quite delicate. Indeed, Scade relies on a strong implementation hypothesis, the so-called "synchronous approach", and it is not obvious how the properties of a model developed with a synchronous approach can be preserved when transferred to another setting, say for example the more "hybrid approach" favored by languages such as Simulink. To state this problem simply: we may always translate a piece of text from one language to another, but it is difficult to prove that the original meaning is preserved in the process. We sketch an example (see Sect. 1.3) of the kind of "semantic" problem that can arise in the following sections.

[1] In the remainder of theis text we will sometimes use "ETCS specification" instead of the longer form "SRS subset 26 specification".

The rest of the report is as follows. In the remaining sections of chapter 1, we briefly describe the Scade language and the particularity of the OBU Scade model. In chapter 2, we list the criteria that should be taken into account when evaluating the possible migration scenarios. Each migration scenario can be described by a transformation from (one concrete syntax for) Scade to another language. This is why, in chapter 3, we describe the existing output formats supported by Scade suite. Finally, in chapter 4, we define possible scenarios based on the tools that can exploit these different output formats and list the strengths and weaknesses of each approach.

1.1 Overview of Scade

Scade is a formal modeling language that provides both a graphical (diagrammatic) and textual concrete syntax. Formal here means that the intended meaning of a model can be defined unambiguously using a mathematical framework. It is therefore possible to reason on models and to prove properties on them with a very high-level of confidence. More precisely, Scade models are synchronously clocked state machines, interacting on infinite data streams, that can be nested and intermixed with each other without limitations. Actually the core of the Scade language is based on Lustre [3, 4], a declarative, synchronous dataflow programming language for reactive systems defined in the early 1980s at the Verimag laboratory in Grenoble.

Another property of Scade is that it is an executable language with a deterministic semantics, meaning that no randomness is involved in the “execution” of a Scade model; A model will thus always produce the same outputs from a given set of inputs. All these characteristics of Scade (formal, executable and deterministic) made it possible to develop the first certified code generators to C and ADA according to the DO-178B and the EN50128. This is a benefit of using Scade with respect to competing frameworks.

1.2 Motivations for the Adoption of Scade in the Project

The Scade language is supported by a toolbox, the *Scade suite*, that provides an integrated design and development environment. Scade suite offers several services: simulation and debugging at the model level; test case execution; model test coverage measurement; code verification on models; . . .

An extension of the tool, called *Scade system*, provides additional functionalities that are of interest in the context of the openETCS project. Basically, Scade System provides systems modeling and model generation based on the SysML standard and the Eclipse Papyrus editor. In particular, Scade system offers “partial” gateway from and to SysML and integration with Eclipse through the use of the Eclipse Modeling Framework (EMF). This is relevant for the openETCS project since Eclipse was selected as the de facto open source platform for integrating the tools developed in the project, while SysML was selected for modeling the architecture of the system.

Unfortunately, the SysML (architectural) model obtained from a Scade specification does not preserve all the information. It only contains information on the states machines and some information at the level of the interfaces; the detailed code related to the functions and the state transitions is left over. Therefore we cannot simply use the output of the Scade system tool for our migration. Also, at the moment, there is no proper way to import a “Scade system” model from a “Scade Suite” model, even though such a gateway was requested by the openETCS project more than a year ago. What is available at the moment is a synchronization between models on “shared” blocks and their inputs and outputs. It means that if you start using Scade System to define the different software blocks, then you can refine those blocks with Scade suite, evolve both models concurrently and ensure synchronization of blocks with their data interfaces. There is no support for models that have been only edited using the Scade suite.

Scade has been used in production on several big projects that covers many application domains: avionics, rail, automotive, . . . It allows the production of rapid prototype as well as of safety related target system software. The intrinsic qualities of Scade are enough to motivate its choice for modeling the OBU functions. Nonetheless there are also important contextual reasons. Most importantly, there does not exists an open source solution with the same degree of maturity. For example, even though the Scade editor requires a moderately high learning curve, it is a solid piece of software that has been tested on many real-size examples. Another circumstantial reason for the adoption of Scade in the project is that some of the partners already had a working knowledge of Scade. Also, many test models related to the ETCS had been produced in the context of our initial evaluation phase and were already available in the first months of the project. Henceforth, considering the strong constraints on the deadlines and the large amount of modeling work involved, Scade was an obvious choice. In any case, it was certainly the choice incurring the less amount of risk for reaching the objectives of the project.

1.3 Associated Drawbacks

We can also list several drawbacks related to the choice of Scade. First, the choice of a synchronous language implies that there is a choice of implementation imposed on the users of the models (namely the choice of a synchronous architecture). This restrict, or at least complicate, some possible choices when implementing the OBU. Indeed, the synchronous hypothesis is well-suited for control functions that operate on a single computer since, in this case, it is not difficult to have all the components share a common clock. Nonetheless, this choice may have some non-trivial consequences if a supplier decide to choose a more distributed architecture, where different functions of the OBU are executed on different, loosely-coupled hardware. This is exactly the kind of problems related to the preservation of the semantics mentioned in the introduction.

We can also mention drawbacks related to the choice of a closed-source solution:

- risk of vendor lock-in: actually this is at the core of our migration problem, since there are no easy way to export a Scade model to another modeling language. Note that, with the latest version of Scade suite, it is now possible to view Scade diagrams without buying the software (but of course it is not possible to edit them).
- perennity of the models: there is a (albeit small) risk that the current Scade toolchain will be discontinued in the future, especially when we take into account the very long service life of railway equipments. We should also mention the problem of *perennity of the model editor*. Indeed, we need to be able to continue editing models even when the tooling is discontinued. This is a known limitation of closed-source solutions with respect to "long term support", since it is not possible to subcontract on the continuation of a tool when sources are not available.
- difficulty to have special needs taken into account by the software editor, or services added to the baseline tools.

1.4 Specificity of the OBU model

The models are made available on the GitHub of the openETCS project and a complete description is given in deliverable D3.5.3 [2]. The specification of the OBU functions (`https://github.`

com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions) is a non-trivial example of system modeled with Scade; it has 13 main sub-components each containing more than a dozen Scade diagrams. All the facets of the Scade language are used but some characteristics of the OBU models are worth mentioning:

- The models use a very large number of (complicated) types that corresponds to the large number of variables and values defined in the ETCS specification. For instance there is a large number of "datagrams" defined in the sections related to the radio communication. Therefore we need to target a language with custom data type definition.
- There is a heavy use of state machines in the specification, see for example the diagrams in the ManageLevelsAndModes component[2]. This means that the OBU model is strongly hierarchical and therefore we need to target specification languages that are component-based or. at least, strongly modular.
- The Scade models make use of most of the Scade v6 operators, like iterators on arrays or "safe state automata" for instance. Iterators are a recent extension to the Scade language that are not supported by other flavors of Lustre. In particular, this means that we cannot use a direct translation from Scade to Lustre for our needs.

[2] https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLevelsAndModes/Modes

2 Overall Goal of the Transformation (What do we Want)

Solving our migration problem raises several issues, for instance concerning the simplicity of the migration process or the coherence of the resulting models. Concerning the first point, we are of course interested by methods that are fully automatic and, if possible, at least partially available (implemented). Based on the remarks provided in the introduction of this report, we can try to establish a list of the criteria that can be used to evaluate our migration scenarios.

Open-Source Solution we want to target an open-source solution, or at least a solution that fits in an open-source ecosystem of tools and languages. This is the core of the problems raised in this deliverable.

Semantic Preservation we want to loose as few information as possible during the migration (in the best case we aim at a transformation that is close to round-trip conversion). We also need a method for checking that the semantics of the models are preserved during the transformation.

Simplicity we need a solution that is already existing or at least partially completed. The target language should have all the available tooling: editor, simulator, formal verification, . . . As much as possible, the conversion should be automatic.

Maintainability migration raises also several questions related to the life cycle of the model, that is its “evolvability” in the future and its maintainability. Indeed a model is never a finished object, it is evolving and should be updated. Therefore we want a format that can be easily updated. This means that the structure of the resulting code and the meaning of the identifiers should be easy to understand.

Code Generation we need to have code generation tools, since the current Scade model is already used to generate code for prototyping. The generated code should be easy to verify.

Integration we need an easy integration into the openETCS toolsuite; which means integration into Eclipse and, preferably, gateway to and from SysML through the Papyrus editor

Certification we need to take into account the issues related to the certification of the tools (since the openETCS project ultimately aims at EN50128 certification)

Traceability we need to preserve traceability information. At the moment, most of the traceability information contained in the Scade models are annotations in the comments that refer to relevant sections in the Subset-026 specification. In some parts of the model (e.g. `Manage_TracksideInformation_Integration` and `calculateTrainPosition`), traceability is also supported using the SCADE requirements gateway.

3 Supported Output Formats from Scade

In this chapter, we study the different file formats available from the Scade suite tool. Each format, both textual and graphical, corresponds to a concrete syntax for Scade. This information is interesting since, for each format, we can find tools that can import Scade models that use this syntax; these tools give us possible scenarios for exporting Scade models. In the next chapter, we define our migration scenarios as translations from one concrete syntax of Scade to the concrete syntax of another language.

3.1 XSCADE

Xscade is the basic interchange format used in Scade suite. This is an XML format keeping all editor-specific information; that is both the information on the actual Scade code, but also the positions and the properties of the graphical elements in a diagram (see Sect. 2 of the Scade user manual, `TechnicalManual_SC-TM-R16`, for a description of this format)[3].

Since the last version of Scade suite, it is possible to view a .xscade diagram without buying a license (a software token). Some partial information on the diagrams can also be viewed from the Scade System tool, which means that there should be an EMF description (an `ecore` file) for at least a subpart of the Xscade format (namely the part dealing with state machines). It is not possible to predict whether ANSYS will be willing to freely distribute a separate Eclipse plugin for viewing Scade diagrams or a separate Java model API for manipulating the associated EMF models. Nonetheless Scade System provides capabilities to develop specific export functions using OCL, the TCL scripting language and a specific Java model API.

There exist a few open source tools able to import Xscade files. Nonetheless all these solutions are based on “retro-engineered parsers”, therefore it is not clear if all aspects of the language are supported and if these tools can withstand future evolutions of Scade Suite. For example, we can find an Xscade importer on: (1) the OSATE plugin, an Eclipse plugin for the AADL language; (2) the S3 tool by Systerel[4].

OSATE 2, see `http://www.aadl.info/`, is an Eclipse plugin that provides an importer from XSCADE to AADL. Each component in a diagram is mapped into an AADL system component; Scade component connections are mapped using AADL event data port and state machines are translated into AADL behavior state machine. We give some information on S3 in the next chapter.

Solutions based on retro-engineered parsers are clearly subject to risks if we want to use it concurrently with Scade modeling. This approach is best suited to “one shot” scenarios in which Scade tools are not used anymore once the models are exported. If not, we face the risk of needing to maintain those parsers when the Scade tools evolve.

[3] unfortunately the Scade manuals are not freely distributable.

[4] The S3 tool is not open source.

Advantage: this is the most interesting format for conversion since no information is lost.

Disadvantage: the format is complex and not very well-documented (the related XML Schema or ecore descriptions are not provided); no guarantee that retro-engineered parsers can be used on future versions of Scade.

3.2 Textual Scade format

SCADE Suite can also generate a `.scade` file from a diagram (by default this file is named `kcg_xml_filter_out.scade`) which contains an equivalent of the XML Scade format into a structured textual format. This format is also sometimes referred to as the `kcg` format, since it is the input format used in Scade's code generation tool. This textual format is very close to Lustre code and it is quite low-level.

While the kcg format is very close to Lustre code, Lustre is not a normalized language and the existing implementations have slightly different syntax. For instance, Scade extends Lustre with new operators, like iterators, that cannot be easily translated. Also, during the serialization of Scade code, the names of ports and variables came become quite obfuscated; therefore the resulting code is quite verbose and difficult to edit. Also, the description of nested state machines are flattened and some information on the "hierarchical structure" of the model are lost. Therefore a solution directly based on this format would failed the criteria of *maintainability* listed in the previous chapter.

There are some works that have addressed the problem of converting Scade's version of Lustre to other format. Most notably, Rockwell Collins has developed a proprietary "translation framework" [5] under a project sponsored in part by the NASA Langley Research Center, that provides highly optimizing translators from MATLAB Simulink and SCADE Suite to model-checking tools and theorem provers. But this work is not distributed, even commercially. It is also possible to find some open source tools that provide a "parser" for .scade files, for example in the Scade2B project (`https://github.com/cercles/scade2b`). There also exist a lot of converter from Lustre to other formats that could be adapted to conform to the Scade language.

Advantage: no information loss except for the graphical elements. This format is easier to parse than the Xscade XML format. There are many open-source Lustre implementation available (but none matching exactly the kcg format).

Disadvantage: we loose information on the placement of graphical elements that may be needed if we target a graphical format; readability of the model is poor when generated from a Xscade file.

3.3 Executable Code (C or Ada)

It is possible to obtain C or Ada code using the code generation tools. As such, the resulting code is even less maintainable than the Scade textual format. Nonetheless, we could imagine using part of this output in another modeling language.

Advantage: many open source solution are available

Disadvantage: difficult to update; too specific

3.4 SysML

The SCADE System tool provides support for developing Scade models from a structural descriptions made of SysML Block Definition Diagram (BDD) and Internal Block Diagram (IBD). The full software description—that is the precise behavioral description of each block—can then be designed using SCADE Suite or in the form of manually developed source code. Therefore it is possible to obtain an architectural description of the OBU models in SysML, but the resulting diagrams only contains information on the functions and their interfaces; the detailed code related to the functions and the state transitions are missing and should be added separately.

A possible solution to remedy this shortcoming could be to include the "missing" behavioral description inside SysML blocks, either as kcg or as C code. To do this, it could be possible to use the scripting capabilities of Scade Suite. Indeed, the SysML diagrams generated by Scade system can be queried and transformed programmatically using several methods: the Object Constraint Language (OCL), a query language for EMF-based models standardized by the OMG; the TCL scripting language; or a proprietary Java model API. We are not aware of any previous research project where a comparable scenario has been experimented with.

The gateway to SysML (and more generally to Eclipse and the Papyrus editor) offers the best scenario for current integration with the openETCS toolchain. The existence of these gateways is the reason why Esterel Technologies often claim that the use if the Scade tool-sets are not incompatible with open source projects.

Advantage: best possible integration with Eclipse; possibility to rely on the expertise of the CEA, which is a partner in the project.

Disadvantage: the SysML gateway is fragile (susceptible to break or to change) and not as mature as the other Scade suite tools

4 Possible Scenarios

We identify two main scenarios based on the different tools and languages mentioned in chapter 3. In each case we give a brief evaluation using the criteria defined in chapter 2.

4.1 Conversion to Lustre (and variants)

Our first scenario relies on a transformation from the textual Scade format to Lustre (see Sect. 3.2)

The Scade language originates from (and still has some strong ties with) the synchronous language Lustre. The Lustre language is still being maintained by the Synchrone team at Verimag. The last update to the Lustre V6 version is open source and has been released in 2014 (`http://www-verimag.imag.fr/Lustre-V6.html`). Open source tools for Lustre includes a front-end compiler (`lus2lic`) and a back-end compiler to sequential, deterministic, C code.

Another open source implementation of the Lustre language can be found in the `prelude` tool [6], developed at ONERA. Prelude is a high-level language built upon Synchronous Languages (such as Lustre) and inherits their formal properties. The language extends the syntax of Lustre and provides more high-level constructs, in particular when defining multi-clocks systems. The prelude compiler, called preludec, generates synchronized multi-task C code, independent of the target OS, that can execute on either mono-core or multi-core architectures.

As mentioned in Sect. 3.2, the main problem with this approach is the maintainability of the models since the generated code may be difficult to understand. Another problem is the necessity to develop a transformation from the Scade format to one of the academic Lustre syntax.

Evaluation		
Semantic Preservation	GOOD	we deal with two formal languages that are very close to each other
Simplicity	GOOD	
Maintainability	BAD	we can mitigate the problems with identifiers by adapting the data dictionary but the absence of notation for state machines in Lustre is a big drawback; we loose the graphical representation
Code Generation	GOOD	even better options than with Scade
Integration	MIXED	no existing direct integration with Eclipse but no obvious roadblocks
Certification	MIXED	no certified transformation
Traceability	MIXED	there are no specific support for traceability in Lustre

4.2 Extension of SysML

Our second scenario corresponds to the one described at the end of Sect. 3.4, where we proposed to extend the SysML (architectural) model with behavioral information extracted from Scade.

The idea is to add behavioral blocks to every function in the SysML model that corresponds to the related Scade function.

There are several possible choice for the “content” of these blocks (the language used for the behavioral specification): Lustre code; C code obtained from code generation; fUML; HLL code; . . .

An obvious choice for adding behavioral block description to SysML would be to rely on the “Executable UML foundation”, fUML, that is an OMG standard. fUML is an executable subset of UML that can be used to define, in an operational style, the structural and behavioral semantics of systems.

A more concrete choice is provided by HLL, the intermediate format used in the S3 tool developed by Systerel [7, 8]. S3 is a formal verification tool that provides a gateway from the Xscade format. The language is typed and provides higher-order construction such as a notation for state machines. Nonetheless some work would be needed in order to better take into account (nested) state machines and to adapt the language for its embedding inside a SysML block. The advantage of this solution is that we could benefit from the expertise of Systerel, that is a partner in the project, and that HLL has already been used successfully on the openETCS Scade model, see for example the Systerel VnV user story at `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/04-Results/e-Scade_S3/Scade_S3_VnV.pdf`

Evaluation		
Semantic Preservation	MIXED	we need to choose a “semantics” for SysML (since different tools may have different interpretations of the language, like for examples transitions in State Diagram)
Simplicity	MIXED	While there are plenty of support for SysML, there are no tools supporting an ad-hoc extension with behavioral blocks
Maintainability	MIXED	need to be evaluated (but seems better than with a pure Lustre solution)
Code Generation	BAD	no existing solution off-the-shelf
Integration	MIXED	no existing direct integration with Eclipse but no obvious roadblocks
Certification	BAD	no existing solution
Traceability	GOOD	with this solution we have very strong links between the architectural and detailed design models. Possibility to use the Eclipse traceability standards all the way.

5 Conclusion

It should not come as a surprise that there is no existing solution that satisfies all of our needs. There are some reasons for this situation that goes beyond the simplistic argument of vendor lock-in: first, there is a relatively small user-base for Scade (and therefore less incentive for third-parties to develop export tools); next, there is a lack of compelling scenarios for exporting Scade models to another format, except for code generation. Indeed, Scade targets mainly the detailed design phase (the latter design step of the traditional development cycle) and therefore Scade models have no use after the code generation or model verification activities, which are both supposed to be performed using the Scade tools.

In this report, we propose two plausible scenarios for migrating away from Scade and give some references on projects that already provide import methods from Scade. Unfortunately none of these projects provide a solution that could be used directly and some software development is therefore needed.

Another possible scenario—that is not reported in this deliverable but that was temporarily evaluated—is to consider a migration to a hybrid modeling language, such as Simulink. Actually, considering our goals, a more appropriate target would be Scicos/Xcos (`http://www.scicos.org/`), that provides an open-source alternative. Simulink is a de facto standard when working on heterogeneous models, especially from the point of view of the control and command engineers. There are some reasons against a shift to a Simulink-like formalism. For instance, Simulink or Scicos are mainly used to model and simulate systems, but rarely used to implement critical software[5]. This is mainly related to the fact that it is very difficult to ensure the consistency of the models with respect to the run-time behavior of the generated executable code. These problems could be mitigated by restricting to a formally defined subset of Scicos. Nonetheless, this solution was ultimately put aside since there are no tools available for exporting to Simulink or its open-source. Also, and more importantly, this migration scenario was the one with the more risks associated to it and the one that required the most implementation efforts.

[5] In the domain of high-integrity software, even if initial models of the system with its plant are often described with Simulink in the initial phases, software is more often reimplemented in Scade or imperative code.

References

[1] ERA. *System Requirements Specification, SUBSET-026*, March 2012.

[2] Peter Mahlmann et al. *Deliverable D3.5.3: openETCS Architecture and Design Specification*, September 2015. `https://github.com/openETCS/modeling/tree/master/openETCS%20ArchitectureAndDesign/D3.5.3`.

[3] N. Halbwachs et al. The synchronous data flow programming language lustre. In *Proc. IEEE*, volume 79.9, 1991.

[4] Nicolas Halbwachs. A synchronous language at work: the story of lustre. In *Third ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE)*, 2005.

[5] Michael Whalen, Darren Cofer, Steven Miller, Bruce Krogh, and Walter Storm. Integration of formal analysis into a model-based software development process. In *Proceedings of the 12th International Workshop on Industrial Critical Systems (FMICS)*, 2007.

[6] Julien Forget, Frédéric Boniol, David Lesens, and Claire Pagetti. A Real-Time Architecture Design Language for Multi-Rate Embedded Control Systems. In *25th ACM Symposium On Applied Computing*, pages 527–534, Sierre, Switzerland, March 2010.

[7] Systerel. *Systerel Smart Solver*. `http://www.systerel.fr/en/products/systerel-smart-solver/`.

[8] Marielle Petit-Doche, Nicolas Breton, Roméo Courbis, Yoann Fonteneau, and Matthias Güdemann. Formal verification of industrial critical software. In *Formal Methods for Industrial Critical Systems - 20th International Workshop, FMICS 2015, Oslo, Norway, June 22-23, 2015 Proceedings*, pages 1–11, 2015.