D1.1	**State-of-the-Art**
Access[1]:	**PU**
Type[2]:	**Report**
Version:	**0.3**
Due Dates[3]:	**M12, M24, M36**

Open Cyber-Physical System Model-Driven Certified Development

Executive summary[4]:

This document provides a “State-of-the-Art” overview of the fields in which the ITEA3 research project OPENCPS is performed. The goal of OPENCPS is to provide an open-source integrated framework for co-simulation based on UML, Modelica, and the Functional Mock-up Interface (FMI) standard. The innovations technically focus on UML/FMI/Modelica interoperability, debugging, and efficient simulation.

[1] Access classification as per definitions in PCA; PU = Public, CO = Confidential. Access classification per deliverable stated in FPP.
[2] Deliverable type according to FPP, note that all non-report deliverables must be accompanied by a deliverable report.
[3] Due month(s) according to FPP.
[4] It is mandatory to provide an executive summary for each deliverable.

Deliverable Contributors:

	Name	Organisation	Primary role in project	Main Author(s)[5]
Deliverable Leader[6]	Lena Buffoni	Linköping University	Associated Project Coordinator	X
Contributing Author(s)[7]				
Internal Reviewer(s)[8]	Peter Fritzson	Linköping University	Scientific Coordinator	
	Magnus Eek	Saab	Project Coordinator	

Document History:

Version	Date	Reason for Change	Status[9]
0.1	15/11/2016	First Draft Version	Draft
0.2	21/11/2016	Released Version	Released
0.3	20/11/2017	Second Year Version, minor updates	Released

[5] Indicate Main Author(s) with an "X" in this column.
[6] Deliverable leader according to FPP, role definition in PCA.
[7] Person(s) from contributing partners for the deliverable, expected contributing partners stated in FPP.
[8] Typically person(s) with appropriate expertise to assess deliverable structure and quality.
[9] Status = "Draft", "In Review", "Released".

CONTENTS

ABBREVIATIONS

List of abbreviations/acronyms used in document:

Abbreviation	**Definition**
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
M&S	Modelling and Simulation
SotA	State of the Art

1 OVERVIEW

The OPENCPS project aims to provide an industry grade FMI master simulation tool based on a Modelica-UML compatible run-time system including extending the FMI standard to allow improved co-execution and co-simulation of FMUs generated from Modelica and UML. Another goal is to increase the efficiency and quality of verification, validation, testing activities. In order to provide an industrially exploitable, efficient platform, improvements over the existing state-of-the-art will be made in the fields of state-machine debugging and validation, efficient multi-core co-simulation and model federation.

2 STATE-OF-THE-ART (SOTA) ANALYSIS

2.1 Model-driven development environments

- Dominating modeling and simulation solutions are proprietary, reducing uptake and spread of M&S and model-driven rapid development technology in industry and society.
- The dominating software modeling formalism (UML) works only for software. Recent SysML extensions are not well developed for physical system modeling. Requirements capture is often informal, text-based, leading to inconsistencies and incomplete models.
- ArCon (Architecture Conformance Validation Tool) is an Eclipse plugin for automatic model (in model driven software development) inspection against defined architectural rules.
- Dominating block-oriented modeling tools such as Simulink mainly support a block-oriented causal modeling style which is cumbersome and error prone for physical systems, but fits control modelling well.
- ModelicaML is a UML profile that enables using UML diagram notations for modeling complex physical system and using Modelica for simulations. Moreover, it supports a method for model-based design verification. This method and the ModelicaML prototype were developed by Airbus Group Innovations and Linköping University (see https://openmodelica.org/modelicaml/) in the ITEA OPENPROD project and slightly extended in the ITEA2 MODRIO project.
- In order to compose simulation models automatically, i.e., to combine the formalized requirements system design models and scenario models, a bindings concept was elaborated in (Schamai W. , Ph.D. thesis, 2013) and prototyped in the ModelicaML language.

2.2 Model Federation

Today, the strong division between different domains' expertise involved into the design and development of complex systems causes technological incompatibilities and difficulties. Those are the source of misalignment and friction when data is both conceptually and technically captured by several tools. This problem is generally managed in an ad-hoc manner and its resolution leads most of time in a misuse of tools. For example, the engineering of radar systems involves the definition of a system architecture and signal processing algorithms. The definition of an architecture is usually performed using an in-house tool that supports the vocabulary of the domain (Phased Array Antenna, Pulse Compression, Radar

Cross Section, etc.) while the definition of the algorithms use mathematical tools providing ordinary differential equations (ODE) solvers, differential algebraic equation (DAE) solvers, and matrix operations (multiplication, inversion, etc.) like Matlab. In this context, relying only on unified languages like UML and its derivatives (SysML, MARTE, etc.) is not satisfying because this requires contortions from both system architects and algorithm experts. Indeed, the projection of domain concepts onto a general purpose modeling language is not easy since the mapping is often not straightforward and imposes semantic restrictions. The necessity to adopt external modeling approaches does not soften the learning curve. Manufacturers need methods for seamless and transparent integration of specialized representations and their tooling at low cost. Beyond data and representations integration, we need to provide a real separation of concerns allowing a consistent reasoning and a better design space exploration. Maintaining a global consistency between the various prisms through which the system is studied needs formal foundations to support consistent reasoning and relevant querying (data mining, views extraction, information inferences). This requires the ability to define and maintain semantic traceability links between the artifacts produced by all the tools within a single modeling space. This includes inner domain traceability (refinement, derivation, versioning) or cross-domain traceability (between artifacts or between artifacts and process activities). For some application systems, we need to establish semantic links between the architecture built with a SysML tool and the algorithms e.g. defined in Matlab or algorithmic Modelica, and to clearly define the role of each model in relation with the intentions of the process stakeholders. Capturing such semantic links is particularly tedious and fragile within a context assuming an open world approach. The challenge is then to trace each decision against requirements in order to support both horizontal (versioning, decomposition) and vertical traceability (refinement). We need a framework to guide decision making when the use case imposes the integration of several heterogeneous and unsynchronized viewpoints generating data friction and inconsistencies. With a federation of models, we need to ensure that federated models can be simulated effectively in a distributed manner in order to support seamless integration and verification of IP's produces by different contractors at different levels of abstraction; to our knowledge, this aspect of the co-simulation is not well addressed by any tool nor by existing standards, in particular when we consider MoC refinement (time and data representation, computation, communication, synchronization) over a set of levels of abstraction.

The Modelica association is currently working on the SSP ("System Structure and Parameterization") standard for specifying more in detail how systems should be connected and parametrized SSP (Köhler et al, 2016). This is a good time to influence the standard if during the course of the project it becomes clear that extensions are needed to support UML-Modelica inter-operability.

Two co-simulation tools have been recently developed. DACCOSIM(Saidi et al, 2016) is a tool developed by EDF for co-simulation of models, it is Java based, which raises several issues when composing FMUs, however has interesting features for distributed co-simulation a collaboration with EDF is being envisaged on this topic. FMICOMposer[10] is another tool similar to OMSimulator, however it does not offer TLM support.

[10] Modelon FMI Composer http://www.modelon.com/products/fmi-tools/fmi-composer/

2.3 Safety related automatic code generation for behavioral modeling

Support for automatic code generation greatly enhances the benefits of a model-based development (MBD) process. However, if the generated code affects safety related functions the additional effort to safeguard the generated code diminishes the initial benefit of MBD. A remedy is to rely on (automatic code generation) tools that are qualifiable (i.e., in some sense verifiable or possible to validate) for the identified use cases. Specialized standards (despite conceptual similarities and shared basis standards) apply for different industrial domains, e.g., ISO 13849: (Machinery Control Systems), ISO 26262 (Automotive), DO-178 (Aircraft), etc.). Only a few tools exist (typically restricted to particular industrial domains) which fulfill the necessary requirements for safety related developments and which support control system modeling in an adequate manner. The available tools on the market are based on discrete-time causal data-flow models (block-diagrams), e.g., the TargetLink[11] code generator for Simulink/Stateflow[12] (Schneider, Lovric, & Mai, 2009) or the Scade Suite[13] that has its roots in the synchronous language community (Beneviste, Edwards, Halbwachs, Le Guernic, & De Simone, 2003).

However, these tools rely on proprietary model formats and they are often unaffordable for small and medium sized enterprises. Their restriction to primarily discrete-time causal data-flow prohibits the direct usage of physical (acausal) models in advanced controllers (a manual conversion of equation based physical models into discretized causal data-flow models is needed).

The safety standards suggest several methods for tool qualification, among them (ISO 26262): increasing confidence from use, evaluation of the development process, validation of the software tool, development in compliance with a safety standard. The degree of required assurance depends on the usage context (criticality of the implemented function, etc.). Still, there remains considerable room for interpretation how to achieve adequate qualification in practice. In practice, the method of “tool validation” has proved to be a working approach to achieve tool qualification. A successful example for tool validation by using a “validation suite” approach for the TargetLink toolchain is reported in (Schneider, Lovric, & Mai, 2009). Another practical approach to comply to safety standards is to use “translation validation”: instead of demonstrating in advance that the complete toolchain produces target code that implements the source model correctly (as in (Schneider, Lovric, & Mai, 2009)), the correctness is demonstrated after a translation run by a subsequent validation phase. This approach is advertised for MathWork’s embedded code generator (Conrad, 2009). A problem of the validation suite approach is that qualification efforts can become prohibitively high for more complex input languages. A problem of the translation validation approach is that the subsequent validation phase may just offload the validation problems to the programmer – especially if formal specifications for the input language and the transformation rules of the code generator are missing or are insufficient.

[11] dSPACE GmbH: TargetLink® Automatic production code generator. http://www.dspace.com

[12] MathWorks, Inc.: Simulink®/Stateflow®. http://www.mathworks.com

[13] Esterel Technologies: SCADE Suite®. http://www.esterel-technologies.com

The complexity of the input language is a crucial factor for the applicability of the "validation suite" method. Therefore some of us (Thiele, Schneider, & Mai, 2012) recently proposed a subset of Modelica deemed suitable for a qualifiable code generator. However, up to now no qualifiable Modelica tool is available on the market.

To reap the benefits of an advanced Modeling language like Modelica and to avoid the drawbacks of being dependent on proprietary, closed-source tools for rapid control prototyping Bosch Rexroth AG has started an effort for a prototype based on open source software. Menager et al. (Menager, Worschech, & Mikelsons, 2014) decided to leverage the OpenModelica compiler and adapt the code generation to their needs. Their results are very encouraging, however in order to fully utilize the benefits of directly using acausal models in safety-related, advanced controllers (such as model predictive control or nonlinear inverse-model based control) substantial further advances in code generation technology, particularly addressing V&V requirements, is required.

2.4 Debugging

Since equation-based object-oriented languages are declarative, such debugging is also somewhat related to work in debugging of (mostly) declarative functional languages, of which some is mentioned below.

In lazy functional languages like Haskell the execution order is hard to understand. Partly for these reasons the concept of the Evaluation Dependence Tree (EDT) tree (Nilsson, 1998) was developed to help the understanding and debugging of such languages. On the other hand, functions in an equation-based object-oriented language like Modelica are similar to functions in a strict functional language where arguments are evaluated before the call and in this respect closer to Standard ML (Milner, Harper, MacQueen, & Tofte, 1997).

Explanation of program execution in deductive systems like Deductive Databases (Mallet & Ducassé, 1999) or Description Logic reasoners (McGuinness, Explaining reasoning in description logics, 1996), (McGuinness & Borgida, 1995), (McGuinness & Silva, 2003) has similarities to our MetaModelica debugger (Pop & Fritzson, 2005) because they generate and analyze proof-trees (or derivation trees).

In the context of dynamic (run-time) debugging of equation-based object-oriented languages, some of us earlier (Bunus & Fritzson, 2003) proposed an automated declarative debugging solution in which the user has to provide a correct diagnostic specification of the model, which is used to generate assertions at runtime. Starting from an erroneous variable value the user explores the dependent equations (a slice of the model) and acts like an "oracle" to guide the debugger in finding the error.

Recently a design for tracing symbolic transformations and operations for an equation-based object-oriented language such as Modelica has been developed and efficiently implemented in the OpenModelica compiler (Pop, Sjölund, Ashgar, Fritzson, & Casella, 2012). It allows tracing the causes of errors and presenting the information in a human understandable form.

Regarding debugging in requirement modeling, requirements are most often modeled in dedicated domain specific languages, although some formal foundations with formalisms such as the Four Variable Model (Parnas & Madey, 1991) have been proposed and extended.

Some efforts to integrate requirements into system modelling tools, have also been made, for instance SysML supports requirement modelling. However, SysML has not yet employed requirements formalized as assertions and equations for automatic requirement testing as in (Schamai W. , Helle, Fritzson, & Paredis, 2010). A model driven design process including automatic requirement testing has been recently developed (Schamai W. , Helle, Fritzson, & Paredis, 2011). A model driven design process including automatic generation of fault trees and a prototype toolchain for requirement modelling and analysis in Modelica has been implemented in OpenModelica (Hossain, Nyberg, Rogovchenko, & Fritzson, 2012).

2.5 Modelica model compilers

Most Modelica model compilers and tools are typically closed proprietary solutions, do not yet support software modeling well, are rather monolithic and hard to extend, have sometimes portability problems caused by informal semantics.

Multi-core simulation has just recently been established in a few Modelica compilers, whereas various theoretical publications on that topic exist. The already available implementations cover either an automated task-graph based parallelization (Walther, Waulrich, & al., 2014), (Elmqvist, Eric, & Olsson, 2014) of the continuous model equations or manual decoupling of submodels e.g. Transmission Line Modeling (TLM) (Sjölund, Braun, Fritzson, & Krus, 2010), (Sjölund, Gebremedhin, & Fritzson, 2013). All presented parallelization approaches focus on the continuous-time model equations. Concepts to support multi-core-simulation for hybrid models with a large number of events or discrete, clocked models do not yet exist.

2.6 Process simulation and plant modelling

The introduction of simulators in process industry has been a lengthy and complex process that is still going on. Simulators have been understood as expensive tools that require considerably special skills. User training for a certain simulation product has been expensive and time-consuming. Thus, there is a definite need to develop entirely new business and technological models to complement the sales of simulator licences and to make the implementation of simulators in industry easier and more effective.

Plant engineering has undergone several changes during the last decades. Computer-assisted methods took the planners away from the drawing tables and planted them in front of computer screens to draw process, automation and structural charts. During the last decade, 3D-plant modelling has brought about yet another new way to design and model a plant. As electronic data management is becoming more efficient, we are currently moving away from document and chart-based design towards plant model-oriented design, where a conceptual model is defined in advance for the plant structures. The plant engineering project then uses this model and transfers the planning data between the various actors in this structural form. The traditional charts or 3D-images are views into the plant model in this approach. Another change in plant engineering is its increasingly networked nature. Participation in the design is more and more global and involves persons from various organisations. This, in turn, sets increasing demands on the design and simulation environment.

Computational and simulation models strive to master both larger and larger entities and smaller and smaller phenomena. For example, the need for integration of the CFD (Computational Fluid Dynamics) method and large-scale flow network solution methods exists, but it is not possible to create the most favourable integration with regard to the user by merely combining existing products. In more general terms, this is a question of the need to combine calculation methods of various levels of detail into operations that are visible as parts of the plant model.

A computational model is usually constructed for a specific purpose. Often the models may also be applied to other purposes, however. For example, a dynamic process simulation model that has been constructed to support design can also be of service in automation testing and operator training as well as in performance analysis and optimisation during operation of the plant. Nevertheless, speaking in terms of software technology, the model is often too tightly linked to its original application environment. In this sense, modular flexibility of components in computational models is becoming an increasingly important requirement. In this way, the computational models developed and used in connection with plant modelling could also be used to support model-predictive control or maintenance in intelligent field equipment in an integrated way. Integration of the computational models of various phases of engineering into the plant model would enable seamless combination and thereby simulation of the operation of the various entities. This would support the introduction of new working methods based on simulation in the design of processes, automation and structures.

3 STARTING TECHNOLOGICAL BASE FOR THIS PROJECT

- Eclipse, the world-leading software development framework, open-source from the Eclipse Consortium.
- Papyrus, an Eclipse-based open source and UML-compliant software design suite.
- The OpenModelica model compiler for Modelica and its associated Eclipse plugin with a Modelica/UML profile; open source from the Open Source Modelica Consortium (www.openmodelica.org) and Linköping University.
- The Simantics integration platform based on semantic data modelling; open source from THTH association (www.simantics.org)
- Modeling and simulation tools, environments, interoperability techniques, and application products from partners.
- Open standards and technologies such as Modelica and FMI from the Modelica Association, UML and OWL from OMG.

3.1 Related collaborative research projects.

Link to previous and/or current collaborative research projects:

Project Name	Cooperative Programme	Time period	Technical Focus	Relationship
MODELISAR	ITEA2	2008-2011	MODELISAR Integrates Modelica and Autosar with the	OPENCPS complements, focusing instead on general

Project Name	Cooperative Programme	Time period	Technical Focus	Relationship
			Dassault Systemes proprietary V6 tool suite, focusing on automotive embedded systems.	interoperability and cyber-physical product development based on open-source. Specifically, FMI-related open-source components and standards from MODELISAR will be used in OPENPCPS.
OPENPROD	ITEA2	2009-2012	Development of an open model-driven development, modeling and simulation (M&S) environment that integrates Eclipse with open-source modeling and simulation tools such as OpenModelica and industrial M&S tools and applications.	OPENCPS focuses on efficient execution of models of software and physical phenomena, including a run-time system supporting a high event rate. OPENCPS extends the approach to certified code generation. Moreover, OPENCPS performs an industrial-strength integration of the open source tools OpenModelica and Papyrus.
POSE²IDON	FP7	2009-2012	Simulations and comparisons of diesel ships, full electric ships & hybrid ships. New electrical architecture including energy recovery has been defined and assessed, and a physical (hardware in the loop) demonstrator has been developed by	Application of OPENCPS methodologies and tools to improve the ECOSIM software and physical hardware in the loop demonstrator, with improved physical modelling and system integration.

Project Name	Cooperative Programme	Time period	Technical Focus	Relationship
			SIREHNA. SIREHNA has developed a multi-physics ship simulator (ECOSIM - emission energy and consumption ship simulator) to assess emissions (CO^2, NOX and SOX) and fuel consumption by simulating a complete ship: mechanical part, electric part, thermal part and associated command control systems, as well as the global energy production which is controlled by a Power Management System.	
iFEST	ARTEMIS	2010-2013	Integration Framework for Embedded Systems Tools. This project aimed to provide a framework to integrate tools for the design, implementation and verification of real-time embedded systems, including life-cycle aspects (versioning, bug-tracking, transformations, etc.). It has contributed to the emerging standard on linked engineering data OSLC.	Federation of models developed within the use-cases and distributed co-simulation.

Project Name	Cooperative Programme	Time period	Technical Focus	Relationship
MODRIO	ITEA2	2012-2015	MODRIO extends state-of-the-art modeling and simulation environments based on open standards to increase energy and transportation systems safety, dependability and performance throughout their lifecycle.	Some MODRIO application modeling results will be the basis for certain models in WP6 in OPENCPS, and integrated into the context of an open source environment. FMI enhancements from MODRIO will be the starting point for OPENCPS work in WP2. Debugging and multi-core simulation results from MODRIO will be further enhanced in OPENCPS WP4 and WP5 respectively.
HPCOM	BMBF	2013-2016	Implementation of various parallelization approaches in the OpenModelica Compiler	Extend Parallel Simulation to discrete models with many events.
TRIBUTE	EU FP7	2013-2017	Application of equation based simulators for buildings in the operation phase.	Focus on relationship between simulators and building controls
INTO-CPS	HORIZON	2015-2018	The aim of INTO-CPS project is to create an integrated “tool chain” for comprehensive Model-Based Design (MBD) of Cyber-	Work on FMI cosimulation done in INTO-CPS will be used to drive the work on FMI in OPENCPS

Project Name	Cooperative Programme	Time period	Technical Focus	Relationship
			Physical Systems (CPSs).	
EMPHYSIS	ITEA3	2017-2020	The aim is to develop a new standard (eFMI: FMI for embedded systems) to exchange physics-based models between modelling and simulation environments	The master simulation tool developed in OPENCPS will be used in this project.

4 BIBLIOGRAPHY

Andreasson, J., & Bünte, T. (2006). Global chassis control based on inverse vehicle dynamics models. *Vehicle System Dynamics, 44*, pp. 321-328.

Beneviste, A., Edwards, S. A., Halbwachs, N., Le Guernic, P., & De Simone, R. (2003). The synchronous languages 12 years later. *Proceedings of the IEEE, 91*(1), pp. 64-83.

Bunus, P., & Fritzson, P. (2003). Semi-Automatic Fault Localization and Behavior Verification for Physical System Simulation Models. *Proceedings of the 18th IEEE International Conference on Automated Software Engineering.* Montreal, Canada.

Conrad, M. (2009). Testing-based translation validation of generated code in the context of IEC 61508. *Formal Methods in System Design, 35*, 389-401.

Elmqvist, H., Eric, M. S., & Olsson, H. (2014). Parallel Model Execution on Many Cores. *10th Int. Modelica Conference.* Lund.

Fritzson, P., Auguston, M., & Shahmehri, N. (1994). Using assertions in declarative and operational models for automated debugging. *Journal of Systems and Software, 25*(3), pp. 223-239.

Jochen Köhler, Hans-Martin Heinkel, Pierre Mai, Jürgen Krasser, Markus Deppe, Mikio Nagasawa 2016 Modelica-Association-Project "System Structure and Parameterization" – Early Insights, The First Japanese Modelica Conferences, May 23-24, Tokyo, Japan

Hossain, M. Z., Nyberg, M., Rogovchenko, O., & Fritzson, P. (2012). Computerized model based functional safety analysis. *Proceedings of MATHMOD 2012 - 7th Vienna International Conference on Mathematical Modelling.*

Mallet, S., & Ducassé, M. (1999). Generating deductive database explanations. *International Conference on Logic Programming.* Las Cruces, United States: MIT Press.

McGuinness, D. L. (1996). *Explaining reasoning in description logics.* Diss. Rutgers, The State University of New Jersey.

McGuinness, D. L., & Borgida, A. T. (1995). Explaining Subsumption in Description Logics. *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (pp. 816-821). Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc.

McGuinness, D. L., & Silva, P. P. (2003). Infrastructure for Web Explanations. In D. Fensel, K. Sycara, & J. Mylopoulos, *The Semantic Web - ISWC 2003* (pp. 113-129). Springer Berlin Heidelberg.

Menager, N., Worschech, N., & Mikelsons, L. (2014). A toolchain for Rapid Control Prototyping using Rexroth controllers and open source software. *10th Int. Modelica Conference*, (p. Sweden). Lund.

Milner, R., Harper, R., MacQueen, D., & Tofte, M. (1997). *The definition of standard ML: revised.* MIT press.

Nilsson, H. (1998). *Declarative Debugging for Lazy Functional Languages.* Linköping University. Linköping: Linköping University Electronic Press Distribution.

Parnas, D. L., & Madey, J. (1991). *Functional Documentation for Computer Systems, Vol. 2.* McMaster University, Hamilton, Ontari, Technical Report CRL 237.

Pettersson, M. (1998). Portable debugging and profiling. *Compiler Construction, 1383*, pp. 279-293.

Pop, A., & Fritzson, P. (2005). A Portable Debugger for Algorithmic Modelica Code. *Proceedings of the 4th International Modelica Conference.* Hamburg, Germany.

Pop, A., Sjölund, M., Ashgar, A., Fritzson, P., & Casella, F. (2012). Static and Dynamic Debugging of Modelica Models. *Proceedings of the 9th International Modelica Conference.* Munich, Germany.

Pope, B., & Naish, L. (2003). Practical aspects of declarative debugging in Haskell 98. *Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming* (pp. 230-240). Uppsala, Sweden: ACM.

Salah Eddine Saidi, Nicolas Pernet, Yves Sorel, Abir Ben Khaled. Acceleration of FMU Co-Simulation On Multi-core Architectures. Modelica Association; Linköping University Electronic Press. Japanese Modelica Conference, May 2016, Tokyo, Japan. Proceedings of first Japanese Modelica Conference, 124, pp.106 - 112, 2016, Proceedings of first Japanese Modelica Conference.

Schamai, W. (2013). *Model-Based Verification of Dynamic System Behavior against Requirements.* Linköping University. Linköping University Electronic Press.

Schamai, W., Helle, P., Fritzson, P., & Paredis, C. (2010). Virtual Verification of System Designs against System Requirements. *Proc. of 3rd International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES'2010), in conjunction with MODELS'201.* Oslo, Norway.

Schamai, W., Helle, P., Fritzson, P., & Paredis, C. J. (2011). Virtual Verification of System Designs against System Requirements. In J. Dingel, & A. Solberg, *Models in Software Engineering* (Vol. 6627, pp. 75-89). Springer Berlin Heidelberg.

Schneider, S.-A., Lovric, T., & Mai, P. R. (2009, April). The Validation Suite Approach to Safety Qualification of Tools. *SAE World Congess*.

Sjölund, M., Braun, R., Fritzson, P., & Krus, P. (2010). Towards Efficient Distributed Simulation in Modelica using Transmission Line Modeling. *Proceedings of the 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (EOOLT'2010).* Oslo.

Sjölund, M., Gebremedhin, M., & Fritzson, P. (n.d.). Paralellizing equation-based models for simulation on multi-core platforms by utilizing model structure. 17th Workshop on Compilers for Parallel Computing.

Tate, E. D., Sasena, M., Gohl, J., & Tiller, M. (2008). Model embedded control: A method to rapidly synthesize controllers in a modeling environment. *6th International Modelica Conference.* Bielefeld, Germany.

Thümmel, M., Otter, M., & Bals, J. (2005). Vibration control of elastic joint robots by inverse dynamics models. *IUTAM Symposium on Vibration Control of Nonlinear Mechanisms and Structures*, (pp. 343-353). München.

Thiele, B., Schneider, S.-A., & Mai, P. R. (2012). A Modelica Sub-and-Superset for Safety-Relevant Control Applications. *9th Int. Modelica Conference.* Munich.

Tolmach, P. A. (1992). *Debugging standard ML.* Diss. Princeton University.

Tolmach, P. A., & Appel, A. W. (1995). A debugger for Standard ML. *Journal of Functional Programming, 5*(2), pp. 155-200.

Walther, M., Waulrich, V., & al., e. (2014). Equation based parallelization of Modelica models. *10th Int. Modelica Conference.* Lund.