2008-01-0742

Using Timber in a multi-body design environment to develop reliable embedded software

Johan Eriksson, Mikael Nybacka, Tobias Larsson and Per Lindgren
Luleå University of Technology, Center for Automotive System Technologies and Testing

ABSTRACT

A major challenge for the automotive industry is to reduce the development time while meeting quality assessments for their products. This calls for new design methodologies and tools that scale with the increasing amount and complexity of embedded systems in today's vehicles.

In this paper we undertake an approach to embedded software design based on executable models expressed in the high-level modelling paradigm of Timber. In this paper we extend previous work on Timber with a multi-paradigm design environment, aiming to bridge the gap between engineering disciplines by multi-body co-simulation of vehicle dynamics, embedded electronics, and embedded executable models. Its feasibility is demonstrated on a case study of a typical automotive application (traction control), and its potential advantages are discussed, as highlighted below:

- shorter time to market through concurrent, co-operative distributed engineering, and
- reduced cost through adequate system design and dimensioning, and
- improved efficiency of the design process through migration and reuse of executable software components, and
- reduced need for hardware testing, by specification verification on the executable model early in the design process, and
- improved quality, by opening up for formal methods for verification.

INTRODUCTION

A major challenge for the automotive industry is to reduce the development time while meeting quality assessments for their products. This calls for new design methodologies and tools that scale with the increasing amount and complexity of embedded systems of today's vehicles [14]. To keep and create competitive advantages parts of the embedded software and hardware components have to be developed in-house, which in turn will raise the need for collaboration with suppliers that enables development without disclosing company specific knowledge.

There is a trend in the automotive industry that electronic system development is going from hardware- to software-based solutions [12]. The advantages are, in general, improved flexibility, and lower cost for development, production and maintenance. With this in mind the engineers need tools that can help them in development of their embedded software. Executable models and automatic code generation can be of great help towards an efficient design flow and avoid time consuming and error prone embedded system programming, see for e.g., the Ptolemy project [1], and commercially available UML based approaches such as Rational Rose RT [2].

Design of automotive systems is a truly multi-disciplinary task, spanning from physics, machine elements, and vehicle dynamics, all the way through computer engineering, control theory, computer communication, to man-machine interaction and lifestyle issues. Hence a design methodology embracing a multi-body systems view throughout specification, (system modelling), verification (simulation, resource- and FME-analysis, etc.) and validation (visualization, testing, etc.), is in high demand.

In addition to this, using a common development interface for collaboration and distributed co-simulation could help the engineers at manufacturer and supplier to

develop their individual components concurrently. Such a multi-body co-simulation environment would allow functionality in terms of software to be developed for a model of the physical system dynamics, and vice versa, allow physical parameters to be optimized under given software behaviour. Furthermore, in such a distributed environment, functionality in terms of software components could be developed and refined concurrently at manufacturers and suppliers.

Attempts in this direction can be found in Paper [15], where a heterogeneous and distributed co-simulation environment is presented. The simulator backplane reads a file describing how the modules are connected and manages the communication between the modules as messages during simulation. One drawback is however that the modules need to be adapted to the simulator, hence do not directly reflect their true implementation.

In this paper we undertake an approach to embedded software design based on executable models expressed in the high-level modelling paradigm of Timber [3]. Timber with its object orientation and inherent real-time capability has show to offer a comprehensive approach to the design of embedded real-time software for controlling mechanical systems [21]. In this paper we extend previous work on Timber with a multi-paradigm design environment, aiming to bridge the gap between engineering disciplines by multi-body co-simulation of vehicle dynamics, embedded electronics, and embedded executable models. Its feasibility is demonstrated on a case study of a typical automotive application (traction control).

BACKGROUND

CO-SIMULATION

There are a number of Multi-body System analysis (MBS) software's that are used for, vehicle dynamic analyses, Hardware-In-the-Loop (HIL) and Software-In-the-Loop (SIL). Some of these are listed below:

- VI-Car RealTime, from VI-Grade
- CarSim™, from Mechanical Simulation Corporation
- CarMaker®, from IPG Automotive
- DYNAware, from Tesis Group
- ASM, from dSPACE
- VDMS with MATLAB®/Simulink®, from Milliken Research Associates and Mathworks respectively.
- Etc.

All of these are applicable when performing co-simulations and when vehicle dynamics are needed in the simulation loop. Due to the increasing speed of the solvers for dynamic vehicle simulation it is possible to simulate the dynamics with smaller step sizes which will in turn increase the accuracy of the SIL and HIL simulations. Using SIL and HIL will significantly speed up the development process for embedded software, as testing can be carried out in a real-life like setting prior to vehicle prototype production.

Past work by the authors when it comes to co-simulation and vehicle validation in general are represented in paper [10], where the ideas behind a future Distributed Real-Time Simulation and Validation framework are presented and discussed. Paper [11] addresses the industry demands on methods and tools for mechatronic vehicle development.

FRAMEWORK

MATLAB® / SIMULINK®

Simulink® is used as a backplane to co-simulate vehicle dynamics, embedded electronics, and embedded software. Simulink mediates between simulator components (in our case Simulink native functions, embedded Matlab (M-functions), and S-functions (such as the Timber simulator(s) and the CarSim™ solver). Figure 1 shows the Simulink co-simulation backplane used in this case study. Simulink offers a familiar user interface and commodity functionality for stimuli, logging and visualization.

CARSIM™ SOFTWARE

To represent the vehicle behaviour the CarSim™ dynamic simulation software for four-wheel vehicles is used. It can easily be connected to Simulink® and represented as an S-function in the same. It is also possible to extend the math model of CarSim™. This is useful when the user want to integrate control systems to the model, modify the vehicle model or change the outputs of the model.

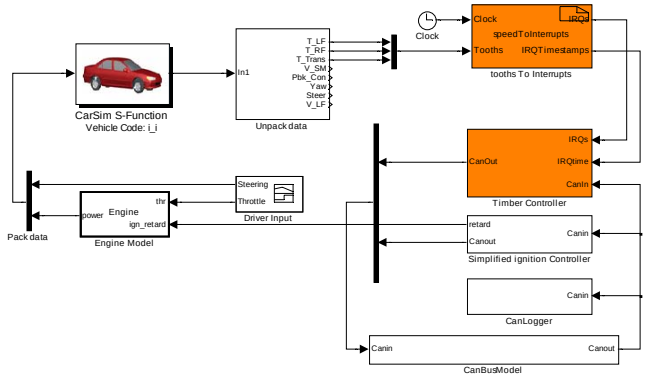

Figure 1. The Simulink framework

TIMBER

Timber adopts the reactive paradigms proven successful to hardware description languages (such as Verilog and VHDL), and offers abstractions of modern functional programming languages, e.g., higher order functions, type inference and objects. The engineering aspects of

Timber are further elaborated in [4] and [21]. Up-to-date information is found online at the Timber development Wiki [22].

Real-Time System Modelling

Timber allows the intended timing behaviour to be straightforwardly expressed directly in the model, as time-bound reactions operating on stateful objects. All realizations that meet the specified timing constraints are “permissible” and to be considered correct, much like that you regard components complying to their data sheets as being acceptable. Using a traditional real-time operating system based approach, timing behaviour is controlled by setting priority levels, either manually or by utilizing some automated approach [22]. In any case, the actual timing behaviour of such a system will be dependent on the load conditions (other executing tasks) and the performance of the target platform, which both may fluctuate over time. Hence, at this stage, from the software description (with its relative priority levels) only a snapshot view of the system’s timing behaviour can be obtained during simulation. However, using Timber, we have the potential to observe and account for the effects of “permissible” timing behaviour of distributed software components, already during simulation.

Object Orientation and Parallelism

Timber objects execute in parallel, (much like you would expect hardware components to live their own life). However, operations on a specific object execute exclusively (forcing other operations on the same object to wait). In this way the state of the object is protected at all times. Traditionally, this is a major obstacle in reactive (real-time) programming, as manually accounting for state protection through RTOS primitives such as locks/semaphores/monitors etc. requires guru-like knowledge [5].

Advanced features of the Timber language

Timber pays heritage to functional languages and comprises a strong type system, featuring type inference, polymorphism, subtyping and higher order functions. Furthermore, the Timber semantics allows for efficient real-time garbage collection[24], thus relieving the programmer from tedious and error prone manual memory management. Altogether, object oriented abstraction, advanced type system, automatic state protection, and real-time garbage collection helps to reduce the risk of hard-to-find programming errors, such as race conditions, dead-locks, memory leakages, and pointer related errors.

System Environment

As in “real life”, the embedded code will reside in, and interface to its environment. For simulation purposes this interface can be emulated by a model of the environment, (eventually to be replaced by the actual hardware). In the early design stages of specification verification, we might settle for an abstract model, and later refine it towards accurately reflecting properties of the hardware realization. With increased refinement, the software model comes closer to the actual production code. In such a way, hardware/software co-design can be carried out to the fullest. Throughout this process, we can during simulation observe and account for the use of shared resources (e.g., CAN busses, memories etc.), such leading us to a feasible partitioning.

Timber and Formal Methods for Analysis

As mentioned, the observed behaviour during simulation express the “permissible” timing defined by the Timber models. Hence, during the design phase we *assume* that execution of the Timber models will meet the timing requirements of the specification. This leaves us with a separate question, namely if execution on a *specific* hardware platform will meet the timing requirements under a given load. Traditionally, this requires excessive testing (by HIL or on real vehicle). However, the Timber semantics is defined to underpin formal methods for verification (such as schedulability [6] and resource analysis [7]), which could be exploited to reduce or even circumvent these time consuming and costly testing activities. Furthermore, formal methods have the potential to actually prove that conditions (such as timing constraints or memory sufficiency) are met under *all* given circumstances. Traditional testing can at best provide evidence under a representative set of cases, (as exhaustive methods are clearly unfeasible). This potential use of formal methods may lead to new liability policies, e.g., a sub-contraction who can show that his subsystem has been formally proven correct for usage according to specification should not be held responsible if used outside this specification (which by proof is the only case where it may actually malfunction).

Timber realizations

For a given target architecture, Timber specifications can be compiled into a subset of C, supported by e.g. the gcc tool suit. The resulting C code is compiled and linked with the Timber run-time kernel (which implements the messaging, garbage collection and scheduling mechanisms), see Figure 2. For a bare-chip target, the executable image will not rely on any additional libraries; hence the system is under full control by the Timber kernel.

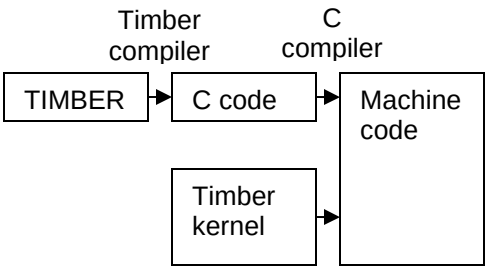

Figure 2. From Timber specification to executable code.

VEHICLE VALIDATION VISUALIZATION

The vehicle validation visualization (v^3) application used in this paper is presented in more detail in [11]. The application is a work in progress to help engineers work and collaborate at distance. The application is written in Java™ and use a Java™ based 3D engine called AgentFX™ [16], it is therefore platform independent and it is also possible to start the application from a web browser via Web Start [17]. Having the possibility to reach the application via the web is a necessity when it comes to distributed collaborative work. We will use the v^3 application in this work as a tool for distributed co-simulation, where the engineers do not have to work at the same physical location to share and view the simulation results, see Figure 3.

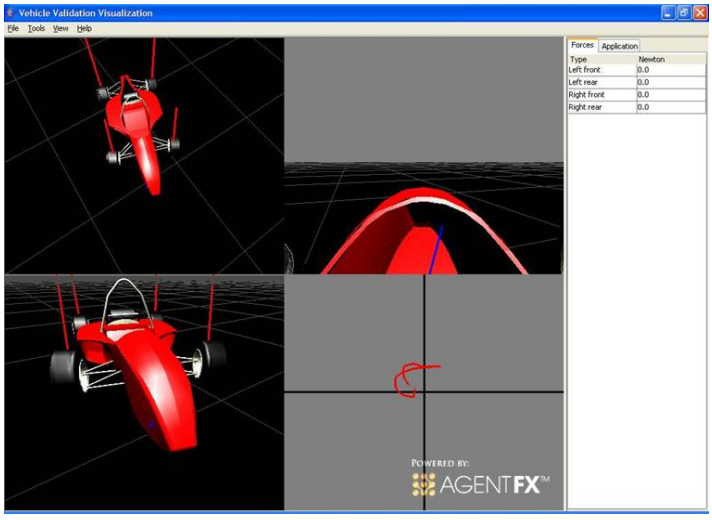

Figure 3. Vehicle Validation Visualization application used during co-simulation.

WORK AND WORK METHODS

CO-SIMULATION

In this paper we propose a design and simulation environment to Timber. By exploiting the unique feature of time-bound reactions, the “permissible” timing behaviour of the system is reviled. Notice that the software description in Timber (with its time-bound reactions) holds a complete view of the system’s timing behaviour, completely free of effects caused by system load and platform performance. In this way software components can be reused in a safe manner in different settings, while retaining their intended real-time behaviour.

Throughout the development process co-simulation has been used to simulate the behaviour of the complete system. Figure 4 shows how the simulation model corresponds to the real system.

Vehicle dynamics

Data to build up a CarSim™ model have been gathered from ADAMS/Car™, so the Kinematics and Compliance (K&C) tests normally performed on real life systems are done virtually in ADAMS/Car™.

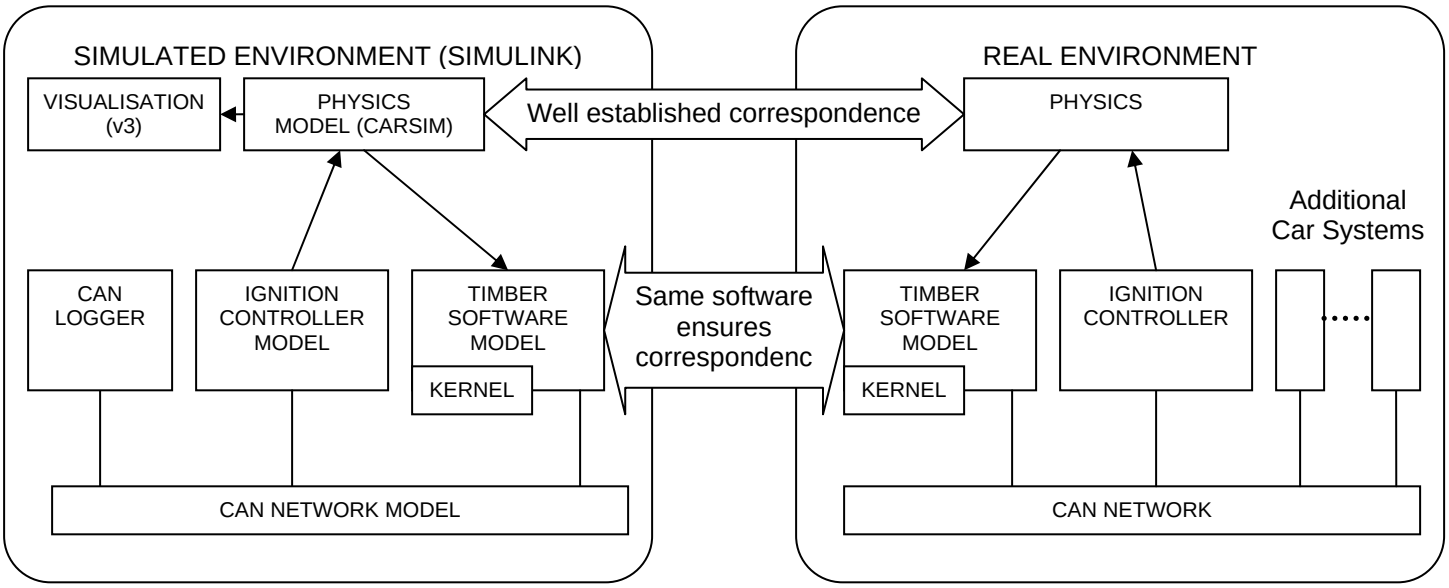

Figure 4. Illustration showing that the same Timber software model is used throughout the design process.

The dynamic model of the vehicle, in this case a Formula SAE car, has been continually developed during years of Formula SAE projects and final thesis works. The moments of inertia have been both measured in real life and obtained from CAD tools.

Development of traction control

The traction controller consists of a number of objects. Figure 6 shows an object view of the system. The operation is in short;

- For every wheel interrupt the `wheelIntHandler()` method is called, if the signal is considered valid it send the new speed to `setWheelSpeed()`.
- The periodical process `calcIgnition()` fetches the slip by calling `getSlip()` and feeds it to `calcPID()`. The result of `calcPID()` is send out on the CAN bus by calling `canSend()`. Then a message is scheduled in 10ms to `calcIgnition()`, (this is how a periodic process is created).

The wheel slip estimator getSlip(),translates the individual wheel speeds into a slip coefficient λ.

$$\lambda = \frac{\left(v_b - \left(v_{fl} + v_{fr}\right)/2\right)}{\left(v_{fl} + v_{fr}\right)/2}, \text{ where } \begin{Bmatrix} \lambda : wheelslip \\ v_b : backwheelspeed \\ v_{fl} : leftfrontspeed \\ v_{fr} : rightfrontspeed \end{Bmatrix}$$

A standard PID module is used to control the ignition retard. The input into this module is the estimated slip and the traction target. The traction target is found by manually looking at a “friction vs. slip rate” (known as $\mu - \lambda$) or “tractive-force vs. slip rate” curve for the tires used. The parameters for the PID controller are found experimentally in the simulated environment.

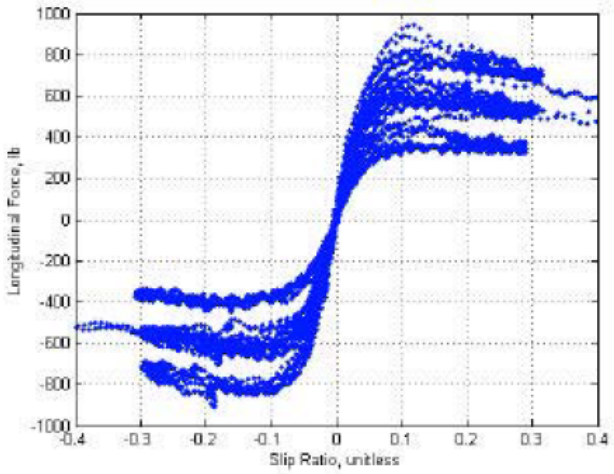

Figure 5. Slip ratio curves in different conditions for a Formula SAE tire [18].

By examining the curve in Figure 5 it is clearly visible that the optimum slip ratio is around 0.1 regardless of the tire load and temperature. The tire data we have used has been obtained from “The Formula SAE Tire Test Consortium“[18].

SYSTEM VERIFICATION

The system verification process has been carried out under the proposed framework, (see section FRAMEWORK). Executable models of the embedded Timber specifications have been designed using Timber-C, a direct programming interface to the Timber Run-Time system.

The system behaviour has been verified (showing stable performance) for the extremes and normal distributions of “permissible” execution windows. The 3D visualization has proven useful for real-time feedback during the development process.

IMPLEMENTATION

The executable Timber models have been compiled for the target platform, a Philips LPC2119 microcontroller featuring a 32bit ARM7 core and 16k SRAM, with 128k flash. The complete hardware platform with the entire car electronic is described in detail in paper [19] and M.Sc. thesis [20].

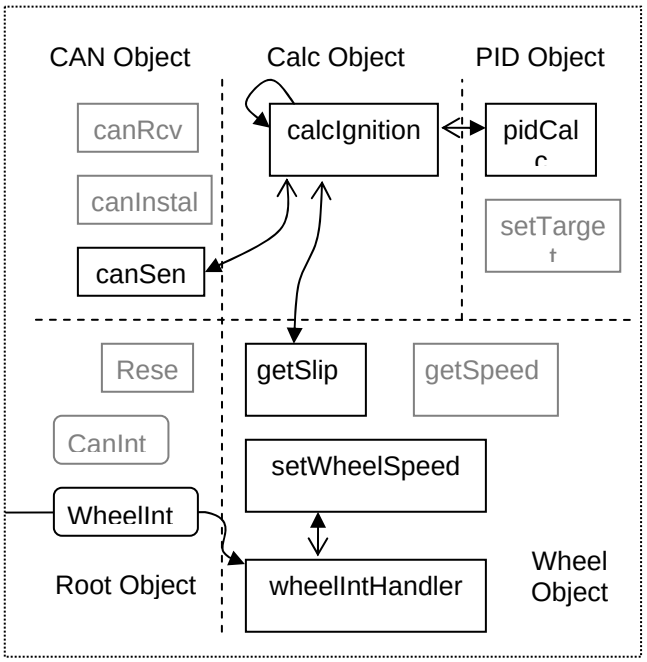

Figure 6. Timber Source

FUTURE WORK

Ongoing and future Timber related research includes;

- finalizing the Timber compiler tool, and
- formal methods and model checking for Timber model analyses, and
- commodity libraries (with examples) for typical vehicular applications, such as controllers, transfer functions, i/o interfaces, loggers etc., and
- visualization of Timber models for specification, simulation and verification.

Ongoing and future work on Distributed co-simulation Visualization;

- a common interface supporting video conference, 3D visualization of vehicle simulation, and
- an environment for collaboration and data distribution for co-operative work.

Future work on real-time test-data transport;

- protocol and wireless technology for real-time test-data transport between vehicle and OEM/supplier,
- sensor network for data acquisition

CONCLUSION

We have demonstrated a multi-paradigm design environment to Timber, bridging the gap between engineering disciplines by multi-body co-simulation of vehicle dynamics, embedded electronics, and embedded executable models. Its feasibility has been shown through a case study of a typical automotive application (traction control), and its potential advantages have been discussed in the following topics:

- shorter time to market through concurrent, co-operative distributed engineering, and
- reduced cost through adequate system design and dimensioning, and
- improved efficiency, through migration and reuse of executable software components, and
- reduced need for hardware testing, by specification verification early in the design process, and
- improved quality, by opening up for formal methods for verification.

ACKNOWLEDGMENTS

The funding from Center for Automotive System Technologies and Testing through Norrbottens Forskningsråd, Calspan and the Formula SAE TTC is greatly acknowledged.

REFERENCES

1. Edward A. Lee, “Overview of the Ptolemy Project”, Technical Memorandum No. UCB/ERL M03/25,

University of California, Berkeley, CA, 94720, USA, July 2, 2003.

2. IBM Software - Rational Rose - Product Overview: IBM Rational Rose Technical Developer [online 2007-02-21] http://www.ibm.com/software/awdtools/developer/technical/

3. A. Black, M. Carlsson, M. Jones, R. Kieburtz, and J. Nordlander."Timber: A programming language for real-time embedded systems." Technical Report CSE-02-002, Dept. of Computer Science & Engineering, Oregon Health & Science University, April 2002.

4. P. Lindgren, J. Nordlander, and J. Eriksson. "Robust Real-Time Applications in Timber". In Sixth IEEE International Conference on Electro,Information Tech, EIT, 2006.

5. Frank Kolnick, The QNX 4 Real-time Operating System, Basis Computer Systems Inc. September, 1998 ISBN 0-921960-01-8.

6. T. P. Baker. "A Stack-Based Resource Allocation Policy for Realtime Processes", IEEE Real-Time Systems Symposium, pages 191–200, 1990.

7. Y. A. Liu and G. Gomez. "Automatic Accurate Cost-Bound Analysis for High-Level Languages", IEEE Transactions on Computers, 2001.

8. Liu C-S., Monkaba V., Lee H., Alexander T., Subramanyam V., "Co-simulation of Driveline Torque Bias Controls", SAE paper 2001-01-2782, 2001.

9. D'Silva S., Sundaram P., D'Ambroslo J., "Co-Simulation Platform for Diagnostic Development of a Controlled Chassis System", SAE paper 2006-01-1058, 2006.

10. Nybacka M., Larsson T., Johanson M., Törlind P. "Distributed Real-Time Vehicle Validation", DETC2006-99154, Proceedings of ASME IDETC/CIE, 2006.

11. Nybacka M., Larsson T., Karlsson T. "Vehicle Validation Visualization", VC_InCo2006_P56, Proceedings of Virtual Concepts 2006.

12. Stensson A., Larsson T., Merkt T., Schuller J., Williams R. A., Mauer L., "Industry demands on vehicle development – methods and tools". Vehicle System Dynamics Supplement, vol. 33, pp. 202-213, Swets & Zeitlinger, 1999.

13. Schäuffele J., Zurawaka T., Automotive Software Engineering – Principles, Processes, Methods, and Tools, SAE Publications, Warrendale, PA, 2005.

14. Müller-Glaser K. D., Frick G., Sax E., Kühl M., "Multiparadigm Modeling in Embedded Systems Design", IEEE transaction on control systems technology, vol. 12, March 2004.

15. Amory A., Moraes F., Oliveira L., Calazans N., Hessel F., "A Heterogeneous and Distributed Co-Simulation Environment", Proceedings of the 15 th Symposium on Integrated Circuits and Systems Design, 2002.

16. AgentFX™, www.agency9.se.

17. Java Web Start, http://java.sun.com/products/javawebstart/index.jsp.

18. Kasprzak E. M., Gentz D., "The Formula SAE Tire Test Consortium – Tire Testing and Data Handling", SAE paper 2006-01-3606, 2006.

19. Eriksson J., Lindgren P., Deventer J., "A Distributed Engine management system for formula SAE", SAE International Detroit 2007

20. Eriksson J. "An Engine management system for Formula SAE", M.Sc thesis at Luleå University of technology, 2006.

21. Eriksson J., Lindgren P "A comprehensive approach to design of embedded real-time software for controlling mechanical systems", 14th Asia Pacific Automotive Engineering Conference (APAC-14)

22. "The Timber Developer Wiki", http://hackage.haskell.org/trac/timber/wiki

23. Saksena M., Freedman P., Rodziewicz P. "Guidelines for automated implementation of executable object oriented models for real-time embedded control systems", Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS '97)

24. Kero M., Nordlander J, Lindgren P. "A Correct and useful incremental copying garbage collector" accepted for publication at The 2007 International Symposium on Memory Management (ISMM 2007)

CONTACT

Johan Eriksson, Ph. D. Student, Luleå University of Technology, EISLAB, SE-97187, Luleå, Sweden. Phone: +46920 491743, Email: johan.eriksson@ltu.se

Mikael Nybacka, Ph. D. Student, Luleå University of Technology, Division of Computer Aided Design, SE-97187, Luleå, Sweden. Phone: +46920 491698, Email: mikael.nybacka@ltu.se

Tobias Larsson, Associate Professor, Luleå University of Technology, Division of Computer Aided Design, SE-97187, Luleå, Sweden. Phone: +46920 493043, Email: tobias.c.larsson@ltu.se

Per Lindgren, Associate Professor, Luleå University of Technology, EISLAB, SE-97187, Luleå, Sweden. Phone: +46920 491092, Email: per.lindgren@ltu.se

DEFINITIONS, ACRONYMS, ABBREVIATIONS

HIL: Hardware-In-Loop

SIL: Software-In-Loop

PID: Proportional Integral Derivative

TIMBER: TIMe reactive emBEdded Real-time