Virtual Integration on the Basis of a Structured System Modelling Approach

Henrik Kaijser, Henrik Lönn
Advanced Technology and Research
Volvo Group
Gothenburg, Sweden

Peter Thorngren
Vehicle Engineering
Volvo Group
Gothenburg, Sweden

***Abstract*—Automotive software and electronics contribute to much of both value and cost for next generation vehicles. Virtual integration and continuous integration and delivery are increasingly used to cut development time, improve product performance and reduce development risk. In this paper, we report on findings from the HeavyRoad project concerning these areas. We elaborate on how modeling patterns for the controller, plant and environment help securing consistency during integration and testing. An architecture pattern as well as a set of views providing appropriate perspectives on the engineering information are introduced. A continuous integration framework is also described. Its intent is to integrate automotive software with models of physical components for the purpose of simulation-based testing. Using these technologies, it was shown how cycle times for software development could be reduced significantly. The paper uses a simple automotive application to illustrate the concepts.**

Keywords—Automotive, Embedded Systems, Continuous Integration, Virtual Integration, Model Based Engineering

I. INTRODUCTION

The authority, criticality and influence of automotive embedded systems increase steadily, while the acceptable development times decrease. Consequently, development methodology must evolve to maintain vehicle quality in spite of shorter lead times and higher product expectations.

Virtual integration is a powerful enabler that allows vehicle and system properties to be analyzed before physical prototypes are available. Continuous development, i.e. iterative and incremental development with comprehensive engineering automation, is another key enabler that provides risk reduction and efficiency through fast and precise feedback on engineering activities.

The HeavyRoad research project addresses both these areas. In this paper, we will describe preliminary contributions related to i) modeling patterns for virtual integration, ii) model viewpoints for automotive integration verification and iii) a continuous integration framework that supports several verification techniques including simulation-based testing of automotive embedded systems.

This work was sponsored by VINNOVA FFI grant 2014-03947.

II. RELATED WORK

Model based systems and software engineering is a mature area, with many notations and tools. Model based software engineering is often synonymous to behavioral modeling and code generation in notations such as SCADE, Simulink or UML. Model based systems engineering relies on architecture representations, to allow properties and requirements to be formulated and assessed in the context of the system description. SysML [9], AADL [2] and EAST-ADL [4] are established architecture description languages with some variations in scope and generality. AUTOSAR [2] provides description mechanisms for software architecture. By using AUTOSAR together with EAST-ADL, system representation from abstract feature content down to software components and its constituents can be represented with semantic and syntactical alignment. In order to support model based integration of model-, software- and hardware-in-the-loop simulations based on the same model, the core architecture descriptions need to be complemented with modeling patterns and model transformations which is addressed in the HeavyRoad project.

Virtual integration is often performed with the purpose to simulate a representation of the system. Behavioral modeling tools typically provide simulation capabilities of components, but there is limited support for system aspects such as execution and communication coordination beyond component level. In such simulations, concurrency, contention and interaction among components are typically not represented. To include these aspects, architecture models can provide system descriptions and a general execution framework can provide for simulations that respect timing, triggering and concurrency aspects.

There are several such of-the-shelf integration frameworks for the automotive domain such as Silver [10], CANoE [11] and Scalexio [3]. However, such tools are mainly intended for interactive, desktop use rather than large scale engineering automation. Further, they are difficult to tailor for company specific needs or refine towards new capabilities concerning work flow, representation, test execution, etc. These issues are investigated further in the HeavyRoad project.

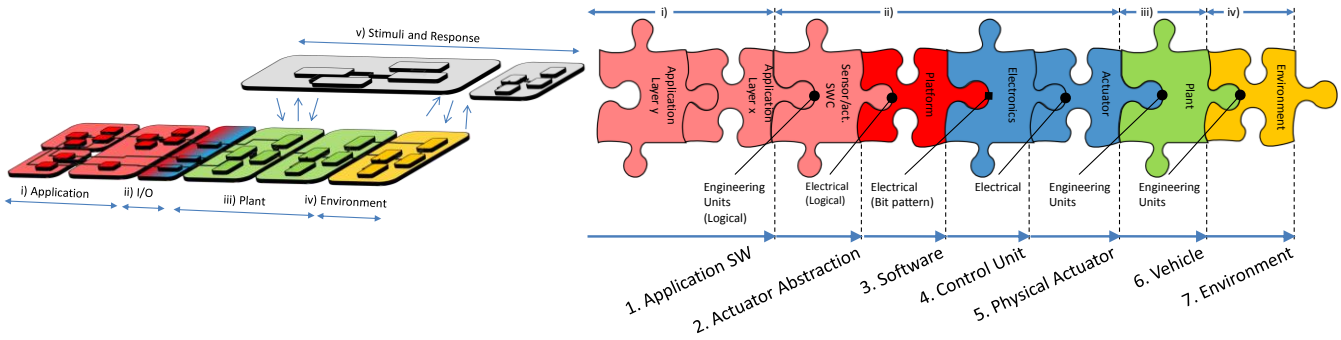

Fig. 1. Modelling Pattern for Embedded System, with interfaces at appropriate locations.

Below we will describe a modelling pattern and a set of viewpoints established in the HeavyRoad project, appropriate for the various simulation and analysis use cases typically pursued during automotive system development. We will then go on to describe the integration and simulation framework developed in the project. Next, we describe how this framework is part of a continuous integration flow. The concepts are then illustrated with an example, before the paper is closed with summary and conclusions.

III. Modelling Patterns

The chosen pattern for the information model is based on the SimArch simulation architecture [12] and extended based on EAST-ADL [4] concepts. We have contributed with a refined representation of structure and behavior and added concepts for plant interfacing, see Fig. 1. The pattern distinguishes between five parts: i) an application part that is typically realized by software but in some cases by models, ii) an I/O part representing sensors, actuators and electrical interfacing, iii) a plant that represents the in-vehicle physical elements, iv) an environment that represents elements outside the ego-vehicle and finally v) the stimuli and expected response for the purpose of specification and verification.

By applying this pattern and maintaining the corresponding interfaces, it is possible to reuse the model parts for different verification use cases. The unidirectional arrows in Fig.1 show suitable interfaces for such experiments. The leftmost arrow (1) represents the boundary of application software corresponding to an engineering units interface for application software-in the loop and model-in-the loop. The boundary may be extended to the electrical interface (2) and thus include the sensor/actuator abstraction. The software boundary (3) includes the platform and is aligned with processor-in-the-loop or virtual targets for target-compiled binaries. The control unit boundary (4) is the electrical interface to sensors and actuators and corresponds to hardware-in-the-loop. The sensor or actuator boundary (5) is the physical interface between the complete embedded system and corresponds to e.g. rapid control prototyping, i.e. where preliminary control systems are used in a mule truck. Finally the vehicle boundary (6) is the external interface of the vehicle, i.e. "tires-to-road".

Examples of how modeling elements are reused in different use cases include running a real control unit together with models of the sensors, plant and environment, or executing binaries on a virtual processor together with models of electronics, and the same models for sensors/actuators, plant and environment. Similarly, models of the embedded system can be connected to sensors and actuators in a real truck and tested in the field.

It should be noted that the granularity can vary depending on the purpose of the experiment. For example, sometimes the I/O part is simplified and not decomposed into sensor/electronics/sensor software while in other cases also the electronics part may be decomposed into wiring harness and discrete electronic components.

IV. Viewpoints and Views

The information model can be considered from a set of five viewpoints [8], see Fig. 2: 1) a logical view collecting functional aspects of software and electronics, 2) an operational view representing components and calibration 3) a plant view representing in-vehicle elements outside the EE architecture, 4) an environment view for elements surrounding the vehicle and 5) an external behavior view. Note that the operational view (2) and the external behavior view (5) have the same structural scope, but the latter concerns the emerging behavior of the complete vehicle.

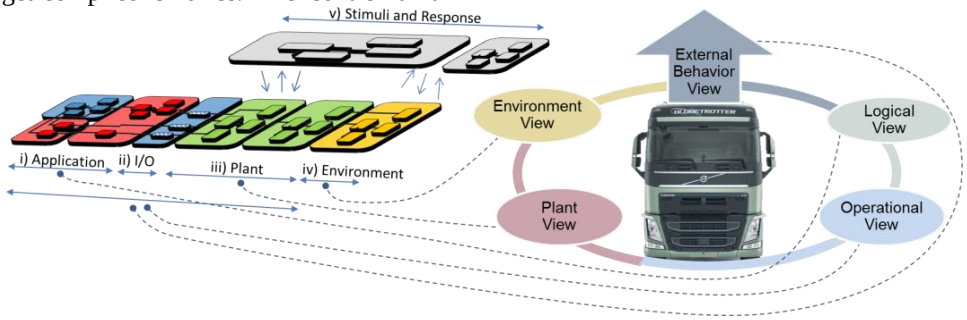

Fig. 2. Information model pattern and Views with correlation relations

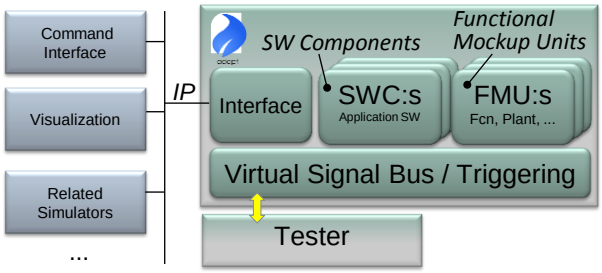

Fig. 3. ADAPT Simulation platform

In the HeavyRoad project, these views have been considered critical to support key roles and process steps during vehicle development. For example, functional testing focuses on the external behavior, design and analysis of subsystems need the logical view, and manufacturing and configuration use the operational view.

V. Integration and Simulation

The HeavyRoad project has specified and developed an integration and simulation framework called Adapt. Adapt can execute AUTOSAR software components together with models of preliminary functions, sensors, actuators, plants and environment. Fig. 3 shows Adapt with a set of modules representing application software, platform, electronics, plant and environment. The interface module exchanges data with external entities e.g. a visualization tool connected over UDP. Modules are triggered by the Adapt framework and exchange data over a virtual signal bus, which can be viewed as a generalization of the AUTOSAR virtual function bus concept.

AUTOSAR application software components are cross-compiled for Windows, but configuration files are taken directly from the production ECU models. The platform software of AUTOSAR is normally emulated to reduce complexity and execution time. In case platform software should be part of the validation, there is a prototype implementation for integrating the open source Arccore AUTOSAR platform in Adapt and let it manage application software, bus communication, i/o, diagnostics, etc.

Non-software parts of the system as well as models of software are represented by Functional Mockup Units, FMUs [5]. The FMU has a description file stating its external interface and an executable part compiled for the host PC. Adapt executes the FMU:s together with the software components and exchanges data over the virtual signal bus.

High execution speed of the simulation is crucial to be able to execute test cases faster than real-time. With a typical experiment setup, the simulation speed has been observed to be an order of magnitude faster than real-time. Some vehicle software were not part of the simulation, but it still gives an indication since only a mid range PC was used.

VI. Continuous Integration and Delivery

Adapt is part of a continuous integration flow currently in use at Volvo Trucks, where engineers provide application code, compiled models and configuration definitions to build servers, which

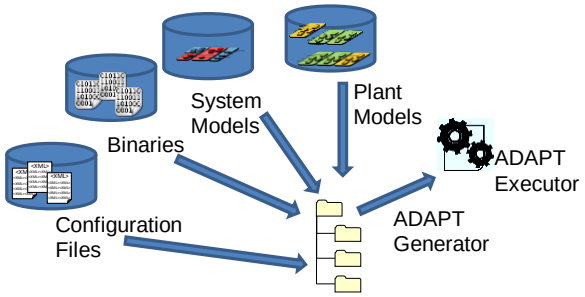

Fig. 4. Continuous Integration Flow

1. Build application software
2. Integrate application software with executable, compiled models of surrounding system elements
3. Run tests on complete system
4. Report back to the engineer

Through this automated flow, engineers get fast feedback on changes and are able to sustain an iterative development style with loop times of a few minutes for the basic verification. These loop times are substantially shorter than what was possible without the continuous integration flow. Because plant models of the physical system are included in the simulations, integration tests cover the complete system from end to end. More elaborate verification and analysis can be scheduled on a less frequent basis, to strike the correct balance between feedback time and verification confidence.

VII. Example

To illustrate the HeavyRoad concepts and tooling described above, we will use a simple windscreen wiper system in a model-in-the-loop scenario. Fig. 5 shows an EAST-ADL model represented in the EATOP framework [5]. The example illustrates the modeling pattern with a functional architecture with wiper controller (i application), wiper stalk, motor and sensor with corresponding electronics (ii I/O part), windscreen (iii plant) and weather (iv environment).

The input from the driver is not part of the model, but provided from the virtual stalk in the visualization framework, see Fig. 6 that captures the ADAPT simulation of controller, plant and environment components. These exchange data over a virtual signal bus, which also transfer data to the interface that communicates with the external visualization and interaction components. The latter uses geometries from the CAD systems, allowing early and iterative visualization of the product behavior. This scenario is real-time and interactive and suitable for explorative purposes in an early phase of development.

Another use case is accelerated simulation of production software. The ADAPT framework then executes the production code of the wiper system together with the same plant and environment models. The test automation part of the integration framework provides stimuli and verifies response.

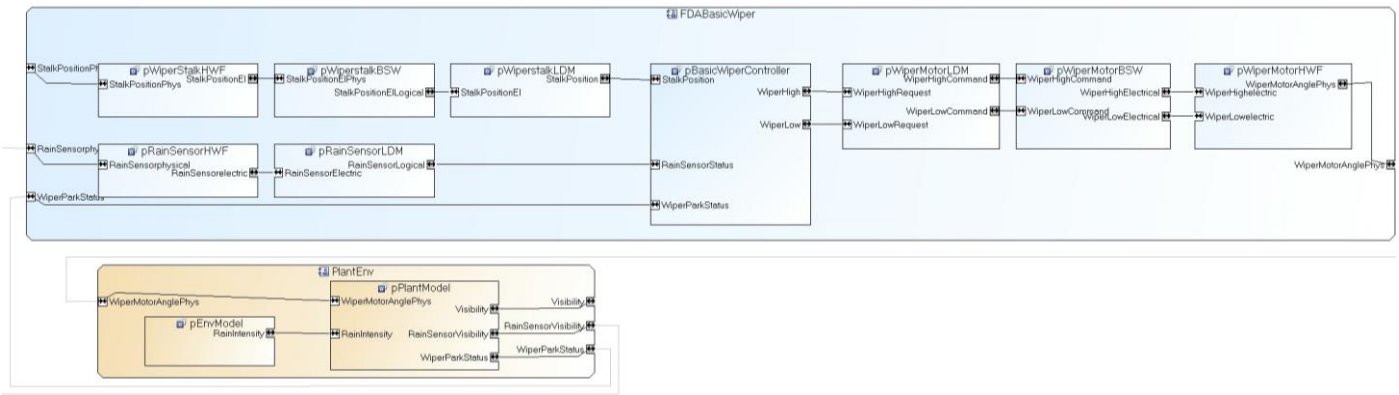

Fig. 5. Functional architecture of Wiper controller and plant

VIII. Summary and Conclusions

This paper has described a modeling pattern and viewpoints suitable for virtual integration of automotive embedded systems. By organizing the system model appropriately, parts of the model can be used for varying purposes ranging from hardware-in-the loop to model-in-the-loop simulation. Further, it is possible to mix simulation constituents, for example different parts of the system may be represented by the production code, models or the real hardware.

We have also described a continuous integration platform that allows virtual integration and verification of automotive software together with models of plant and environment. This is a key enabler for continuous integration and delivery, as meaningful tests can be performed on software before the rest of the vehicle is available. Virtuality is also an enabler for tests that could not be performed on real vehicles e.g. because of safety concerns, instrumentation problems, time consumption, cost or vehicle availability. Using project results we have observed development loop times being reduced to minutes, and verification rate of virtually integrated systems is an order of magnitude faster than with real systems.

Rigor in the representation of engineering information together with the ability to integrate and simulate early and iteratively is a key enabler in meeting future demands on short development cycles.

While many flaws can be efficiently detected with an analysis and simulation-based approach, testing of the embedded system and the complete vehicle is still necessary. Software is simulated in a way that cannot completely avoid inaccuracies in timing and resource handling. Electronics, I/O and mechanical parts rely on models with a limited fidelity and assumptions that may be wrong or inaccurate compared to the real vehicle. These experiment risks are mitigated by validation and calibration, but testing of real systems allows confidence to be increased even further.

References

[1] Architecture Analysis & Design Language, SAE Standard AS-5506

[2] Autosar Development Partnership: AUTOSAR 4.2, www.autosar.org

[3] DSpace Scalexio – product information. http://www.dspace.com

[4] EAST-ADL Association: EAST-ADL 2.1.12, www.east-adl.eu

[5] Eclipse: EAST-ADL Tool Platform. www.eclipse.org/eatop

[6] FMI development group: FMI 2.0, www.fmi-standard.org

[7] HeavyRoad: HeavyRoad web site, www.heavyroad.eu

[8] IEEE: Standard 42010, Systems and software engineering — Architecture description.

[9] Object Modelling Group: OMG Systems Modeling Language, www.omgsysml.org

[10] QTronic Silver – Product Information. http://www.qtronic.com

[11] Vector CANoe – product information. http://www.vector.com

[12] Vinter, Jonny: Simarch Final Report, V-ICT Project SimArch, Sweden, 2010

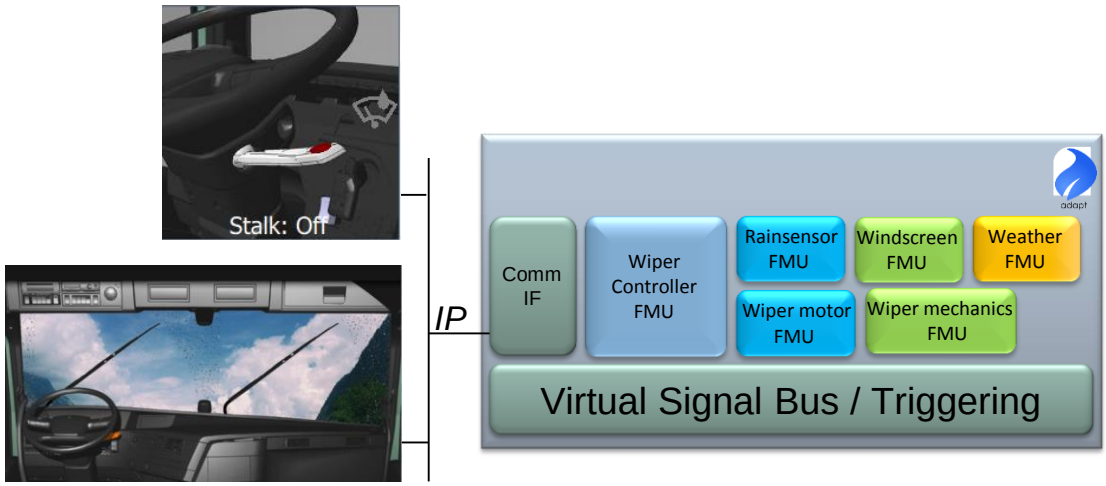

Fig. 6. Adapt wiper system simulation together with visualization tools