An Integrated Framework for Component-based Analysis of Architectural System Models

Raluca Marinescu, Cristina Seceleanu, and Paul Pettersson

Mälardalen Real-Time Research Centre (MRTC)
Mälardalen University
Västerås, Sweden
{raluca.marinescu, cristina.seceleanu, paul.pettersson}@mdh.se

Abstract. Verifying architectural models of embedded systems is desirable, since architecture can impact the performance and resource usage of the final system implementation. To fulfill this need, one could think of combining formal verification and testing to achieve proofs of system correctness with respect to functional and extra-functional requirements. Our first step to accomplish this goal has concretized in the development of a framework that integrates architectural models described in East-adl language with component-based model-checking techniques. The framework is supported by a tool called ViTAL, which captures the behavior of East-adl functions as timed automata models, which can be formally verified in the Uppaal Port model-checker that exploits the components-based semantics at the architectural level. Later, the same formal models will help generate test-suites to provide support for model-based testing.

Keywords: East-adl, V&V techniques.

1 Introduction

Nowadays, many automotive functions are real-time, so a thorough Verification and Validation (V&V) is necessary to ensure real-time requirements at the architectural level. Current V&V tools are working isolated and their interaction is difficult [9], if not impossible. A smart combination of this different techniques, together with their successful application in industrial practice, could be the next step in the V&V evolution.

Lately, a lot of effort has been devoted to generate test-suites from system models (e.g., UML [4], Timed Automata (TA) [7]), and also to verify such models (e.g., Uppaal [1], PROGRESS [10]). However, these are sparse results with regard to the combination of V&V techniques.

The automotive industry provides its system specification in architectural description languages with no precise formal semantic, making it harder to use model-checking tools to analyze such embedded system (ES). In practice, some companies (e.g., VOLVO Technology AB and Continental Automotive) are using East-adl [2], an architecture description language dedicated to automotive ES, which does not provide the possibility to construct, verify, and transform its models using formal techniques. Formal verification of both functional and timed behavior is necessary to ensure the real-time requirements at the architectural level, making EAST-ADL models a good basis for a combined V&V framework.

In our research, we intend to bring closer architectural description languages and verification techniques, through a new framework that consists of an integrated methodology that has been implemented in a tool called ViTAL [1](A **V**er**i**fication **T**ool for EAST-**ADL** Models using UPPAAL PORT) [3], which provides Component-Based (CB) verification of EAST-ADL models via UPPAAL PORT. The tool lets one describe functional EAST-ADL behavior in TA semantics. To show the applicability of our tool and method, we illustrate its use on an industrial prototype, that is, Volvo Technology's Brake-by-Wire system. Later, ViTAL will be extended with test-suite generation capabilities to provide support for model-based testing, by generating test suites corresponding to various functional and extra-functional goals.

The paper is organized as follows. Section 2 briefly overviews EAST-ADL architectural language, UPPAAL PORT model-checker, and model-based testing. Section 3 presents the work already done and some preliminary results. In Section 4 we give a short description of the Brake-by-Wire industrial case study. Next, Section 5 describes our steps to finalize the proposed framework, before concluding the paper in Section 6.

2 Preliminaries

EAST-ADL. The architecture description language EAST-ADL is structured into five abstraction levels, which represent different stages of the engineering process, and provides traceability between them [2]. In addition, the structural organization of EAST-ADL has modeling constructs for behavior, requirements, timing, variability, and safety aspects. It captures structural components that refer to external or internal behavior as Simulink models.

UPPAAL PORT. Based on UPPAAL model-checker an extension for CB systems called UPPAAL PORT was released [5]. It uses local time semantics and a Partial Order Reduction Technique (PORT) to improve the efficiency of the verifier. UPPAAL PORT is suited for the analysis of EAST-ADL models because it assumes a "read-execute-write" component model semantics in its input language.

Model-based Testing (MBT). It derives test suites based on the specified functional requirements from a behavioral model of the system, covering one or more particular criteria [8]. A test harness executes the test suite against the implementation under test and the result is compared to the expected result, prescribed by the specification, by a test oracle. The test oracle delivers a verdict for each test case in the test suite.

[1] ViTAL is available at http://www.vitaltool.org

3 Contribution of the Thesis

The behavior of an EAST-ADL function prototype (FP) is described using external notations such as UML and Simulink, which do not have direct support for formally verifying EAST-ADL models. We propose a framework that integrates architectural description languages and verification techniques for CB ES, which have been implemented in the ViTAL tool. As depicted in Fig 1, the system designer creates the EAST-ADL models in a dedicated tool (e.g. Papyrus) and adds behavior to the EAST-ADL components, as TA models, such that UPPAAL PORT model-checker can be used to simulate the system model and verify various requirements (e.g., functional and timing requirements). We specify the internal behavior of each elementary FP as TA, and construct a complete system behavior model by the parallel composition of local behaviors. In addition, we map FP ports onto UPPAAL PORT read/write variables. A composition of function behaviors is considered a network TA that enables us to analyze and verify behaviors of the entire system using UPPAAL PORT model checker. To be able to perform this, we implement an automatic model transformation to UPPAAL PORT, which enables UPPAAL PORT to handle EAST-ADL models enriched with TA behavior as input.

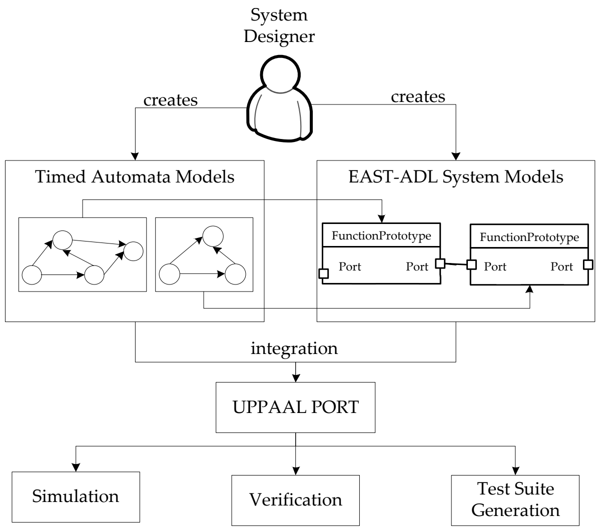

Fig. 1. The workflow of the integrated simulation and verification tool of EAST-ADL models

The above steps are implemented in our ViTAL tool, which provides an integrated environment for architectural description languages and verification techniques, based on different Eclipse plug-ins, as depicted in Fig. 2. The User

Interface integrates an editor for East-adl models in the Eclipse framework, as well as a TA editor to model the timing and functional behavior of East-adl FPs. Uppaal Port introduces support for simulation and verification, using a client-server architecture. The Uppaal Port model-checker consists of two modules: the Eclipse plug-in used as the graphical simulator, and the server executing the verification. Using the integrated simulator it is possible to validate the behavior and timing of an East-adl system model, prior to design and implementation.

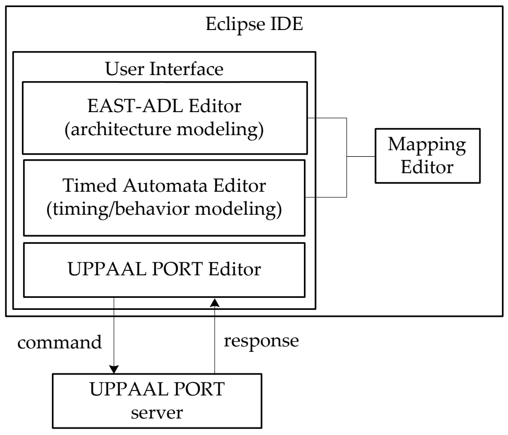

Fig. 2. Overview of the ViTAL tool architecture

In order to integrate the formal model of Uppaal Port TA with East-adl, we need to first perform a semantic anchoring of the latter to a component model that obeys the *read-execute-write* semantics, hence preserving the informal semantics of East-adl without altering its structure. The Mapping Editor shown in Fig. 2 can be seen as a function $\pi : EAST - ADL \rightarrow UPPAALPORT$, which maps each FP to an intermediate component, input ports to intermediate component data-flow input ports, output ports to the intermediate component data-flow output ports, connectors to the intermediate component connections, and the timing constraints to timing annotations.

ViTAL provides support only for the analysis of functional and timing requirements of EAST-ADL functions, but the limited software and hardware resources of complex automotive embedded systems require the analysis and verification of extra-functional requirements. Due to the lack of resource modeling notations in East-adl, allocations of components cannot be analyzed and refined at earlier phases of design. To address this problem, we propose a modeling extension to the current East-adl language with associated abstract resource information [6]. In order to annotate and reason about resource usage of EAST-ADL models, we need a semantic extension of the model and its behavior. At the structural level, the resources are part of our extension of the EAST-ADL language in order to obtain resource awareness. At the behavioral level, Priced Timed Automata (PTA) can be used as a framework for the formal analysis of the corresponding models and the resource consumption represented as real-valued cost variables, and their evolution.

4 Applying ViTAL on the Brake-by-Wire System

The Brake-by-Wire (BBW) system consists of five Electronic Control Units (ECUs) connected by a network bus: a central ECU connected to the brake pedal and another four ECUs connected to each wheel. The central ECU has three components: a brake pedal sensor, a component that calculates the global brake torque from the brake pedal position, and a component that distributes the global torque to the four wheels. Each wheel ECU also has three components: a sensor that measures the wheel speed, a component for the brake actuator, and an ABS controller. The ABS controller is based on a simple logic: if the slip rate of the wheel is larger than 0.2, then the brake actuator is released and no brake is applied. Otherwise, the requested brake torque decided by the central ECU is used.

A set of properties concerning the safety and liveness of the BBW system have been verified with ViTAL. Here, we present a few representative properties that we have verified in our previous work [3]:

– The property of deadlock freedom;
– Timing properties, like the end-to-end deadlines;
– Functional properties, which relate to the slip rate value.

5 Future Work

To provide a real combination of V&V techniques, tailored for EAST-ADL architectural language, which is our main research goal, we plan to extend our framework with offline test suite generation capabilities for both functional and extra-functional testing goals. The tool support will be based on ViTAL and will use model-based testing to derive test-suites from EAST-ADL functions enriched with TA behavior models, by exploiting the trace information resulted as witnesses (or counter-examples) from UPPAAL PORT verification of appropriate properties.

To be able to carry out resource-wise analysis of EAST-ADL models, we intend to integrate our extension with a formal model, that supports resource analysis techniques for performing quantitative consumption analysis. We could show how analysis goals (e.g., feasibility analysis, optimal resource analysis) can be formalized and reasoned about by combining EAST-ADL with an abstract resource-aware behavioral model [6].

Last but not least, we intend to transform the abstract test-suites in executable test-suites and use them on the actual system implementation to be able to asses the effectiveness of our framework.

6 Conclusion

Our research goal of V&V of East-adl models requires a consistent environment that brings together model-driven development, formal analysis, and test-suite generation. The employed formalism is the TA framework that captures the execution flow inside each FP and the complex interactions between components. In this paper, we have described a methodology towards the integration of East-adl and Uppaal Port and its implementation in a tool called ViTAL. As future work, we will extend ViTAL with test-suite generation capabilities to enable the verification of East-adl models. Through our framework, we hope to bring together the V&V techniques, tailored for architectural models of ES.

Acknowledgment: The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement number 269335 and from VINNOVA, the Swedish Governmental Agency for Innovation Systems, within the MBAT project.

References

1. Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Uppaal a tool suite for automatic verification of real-time systems. In *Hybrid Systems III*, Lecture Notes in Computer Science. Springer, 1996.
2. MAENAD Consortium. East-adl domain model specification. http://www.maenad.eu/, 2011.
3. E.P. Enoiu, R. Marinescu, C. Seceleanu, and P. Pettersson. Vital: A verification tool for east-adl models using uppaal port. In *17th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2012.
4. S. Gnesi, D. Latella, and M. Massink. Formal test-case generation for uml statecharts. In *Ninth IEEE International Conference on Engineering Complex Computer Systems*, 2004.
5. John Håkansson, Jan Carlson, Aurelien Monot, and Paul Pettersson and. Component-based design and analysis of embedded systems with uppaal port. In *6th International Symposium on Automated Technology for Verification and Analysis*, 2008.
6. Raluca Marinescu and Eduard Paul Enoiu. Extending east-adl for modeling and analysis of system for resource-usage. In *Proceedings of the 4th IEEE International Workshop on Component-Based Design of Resource-Constrained Systems*. IEEE Computer Society Press, 2012.
7. Brian Nielsen and Arne Skou. Automated test generation from timed automata. In *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science. Springer, 2001.
8. A. Pretschner. Model-based testing. In *7th International Conference on Software Engineering (ICSE)*, 2005.
9. M. Utting and B. Legeard. *Practical model-based testing: a tools approach.* Morgan Kaufmann, 2007.
10. A. Vulgarakis, J. Suryadevara, J. Carlson, C. Seceleanu, and P. Pettersson. Formal semantics of the procom real-time component model. In *35th Euromicro Conference on Software Engineering and Advanced Applications*, 2009.