Architecture-Driven Semantic Analysis of Embedded Systems
Preface to the special issue

Jérôme Hugues[a], Oleg Sokolsky[b]

[a]*Institut Supérieur de l'Aéronautique et de l'Espace, 10 avenue Edouard Belin - BP 54032 - 31055 TOULOUSE CEDEX 4 FRANCE*
[b]*University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104*

Modern distributed real-time embedded (DRE) systems are growing ever larger in scope and complexity. Assuring critical properties of such systems requires system architects to maintain a coherent view of the system throughout the development process, refining the view as the design evolves. It is no surprise, then, that architectural modeling of complex embedded systems has been gaining prominence in recent years, both in academia and in industry. An architectural model represents components in a distributed system as objects with well-defined interfaces, captures information flows and dependencies within the system as connections between ports on component interfaces, and specifies component attributes that can be used in analytical reasoning about the model. Models are hierarchically organized, so that each box can contain another system inside, with its own set of components and connections between them.

For a DRE, timing and resource availability form an important part of the system requirements. Therefore, an architecture description language for embedded systems must describe resources of the system platform, such as processors, memories, communication links, etc. Several architectural modeling languages for embedded systems have emerged in the past decade, including AADL, SysML, EAST-ADL, and the MARTE profile for UML.

Architectural modeling serves several important purposes in the DES development:

- An architectural model allows us to break the system into manageable parts and establish clear interfaces between these parts. In this way, we can manage complexity of the system by hiding the details that are unimportant at a given level of consideration.
- Clear interfaces between the components allow us to avoid integration problems at the implementation phase.
- Architectural models can help with change management, which is critical for keeping the large model self-consistent as the design evolves. Connections between components, which specify how components affect each

other, help propagate the effects of change in one component to the affected components.

- Most importantly, an architectural model can be seen as a repository of the knowledge about the system, represented as requirements, design, and implementation artifacts, held together by the architecture. Such a repository enables automatic generation of analytical models for different aspects of the system, such as timing, reliability, security, performance, etc. Since all the models are generated from the same source, ensuring consistency of assumptions and abstractions used in different analyses becomes easier.

The first three uses of architectural modeling have been studied in the research literature for a number of years [2, 1]. However, the coordination role of architectural modeling in model-based development of DES is just currently emerging. We expect this role to gain importance in the coming years. It is clear that realizing this vision of a "single-source" development process with an architectural model at its core is impossible without having first a clear semantics of the architecture description language.

Semantics of architectural languages and semantics-based transformations of architectural models have been the topic of a Dagstuhl seminar "Architecture-Driven Semantic Analysis of Embedded Systems" held in July 2012. This issue contains a selection of papers that arose from presentations and discussions at the seminar. Seven papers were submitted in response to an open call for contributions to the special issue. Each paper was reviewed by at least three reviewers. At least one reviewer of every paper was not involved with the Dagstuhl seminar, in order to ensure a broad appeal of the selected papers. Following several rounds of reviewing, four papers were selected for the issue. Collectively, these papers cover the typical challenges in developing high-assurance toolchains that support architecture-centric development of DES.

Several approaches exist for assigning semantics to an architectural modeling language. The approach taken by AADL, for example, is to give semantics to each component type in the core language, along with semantics for language annexes that deal with individual aspects of the system such as behavioral specification, error handling, or timing. Semantics of AADL annexes are made compatible with the semantics of the core language during the language definition. MARTE, consistent with the UML philosophy, gives semantics to annotations, such as the clock constraint specification language mentioned below. The paper "DREMS ML: A Wide Spectrum Architecture Design Language for Distributed Computing Platforms" by Balasubramanian and his colleagues takes a somewhat different approach. Their language is viewed as a collection of distinct sub-languages with their own semantics, which are kept consistent by information architecture platform that provides runtime support for systems built with DREMS ML.

At the core of the architecture-driven "single-source" DES design is the automatic generation of analytical models from the common architectural model. Consistency and soundness of the overall design strongly relies on correctness of these model transformations. Bodeveix and his colleagues address the problem of establishing formal guarantees for such transformations in the paper “Towards a verified transformation from AADL to the formal component-based language FIACRE.” The work concentrates primarily on the verification of functional behavior and timing of AADL models.

Timing constraints represent a significant aspect of DES requirements. It is not surprising that two of the articles aim to address verification of timing annotations in architectural models. They approach the problem, however, from quite different perspectives.

The article “Timed behavioral modeling and affine scheduling of embedded software architectures in the AADL using Polychrony” by Besnard and his co-authors addresses timing analysis technique based on timing constraint annotations in the AADL language that target the polychronous model of computation. By contrast, Mallet and de Simone, in the article “Correctness issues on MARTE/CCSL constraints,” target systems based on a looser, asynchronous communication model using clock constraint specification language (CCSL), advocated in the UML MARTE profile.

Hence, we believe this special issue reflects the diversity of modeling and analysis approaches applied to distributed real-time embedded systems. As editors, we thank the reviewers for their diligent work in reviewing, proposing enhancements or clarifications; and to the authors of the selected papers for their contribution. Last but not least, we also thank Dagstuhl for hosting our seminar.

References

[1] P. Cuenot and et al. The EAST-ADL Architecture Description Language for Automotive Embedded Software. In H. Giese, G. Karsai, E. Lee, B. Rumpe, and B. Schtz, editors, *Model-Based Engineering of Embedded Real-Time Systems*, volume 6100 of *Lecture Notes in Computer Science*, pages 297–307, 2007.

[2] P. Feiler, B. Lewis, and S. Vestal. The SAE AADL Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering. In *Workshop on Model-Driven Embedded Systems*, May 2003.