Model Based Architecting and Construction of Embedded Systems

Iulian Ober[1], Stefan Van Baelen[2], Susanne Graf[3], Mamoun Filali[4], Thomas Weigert[5], and Sébastien Gérard[6]

[1] University of Toulouse - IRIT, France, Iulian.Ober@irit.fr
[2] K.U.Leuven - DistriNet, Belgium Stefan.VanBaelen@cs.kuleuven.be
[3] Université Joseph Fourier - CNRS - VERIMAG, France, Susanne.Graf@imag.fr
[4] University of Toulouse - CNRS - IRIT, France, Filali@irit.fr
[5] Missouri University of Science and Technology, USA, weigert@mst.edu
[6] CEA - LIST, France, Sebastien.Gerard@cea.fr

Abstract. This workshop brought together researchers and practitioners interested in model-based software engineering for real-time embedded systems, with a particular focus on the use of architecture description languages, domain-specific design and implementation languages, languages for capturing non-functional constraints, and component and system description languages. Ten presenters proposed contributions on model-based analysis, transformation and synthesis, as well as tools, applications and patterns. Three break-out groups discussed the transition from requirements to architecture, design languages, and platform (in)dependence. This report summarises the workshop results.

1 Introduction

The development of embedded systems with real-time and other constraints implies making specific architectural choices ensuring the satisfaction of critical non-functional constraints (in particular, related to real-time deadlines and platform parameters, such as energy consumption or memory footprint). Recently, there has been a growing interest in (1) using precise (preferably formal) domain-specific models for capturing dedicated architectural and non-functional information, and (2) using model-driven engineering (MDE) techniques for combining these models with platform independent functional models to obtain a running system. As such, MDE can be used as a means for developing analysis oriented specifications that, at the same time, represent the design model.

The MoDELS workshop on *"Model Based Architecting and Construction of Embedded Systems"* brought together researchers and practitioners interested in all aspects of model-based software engineering for real-time embedded systems. The participants discussed this subject at different levels, from modelling languages and related semantics to concrete application experiments, from model analysis techniques to model-based implementation and deployment. The workshop was attended by 61 registered participants hauling from 18 different countries, including six outside of Europe.

2 Reviewed contributions

We received 16 submissions from 8 different countries, of which ten were accepted for presentation. A synopsis of each presentation is given below. Articles [4, 10] are included in this workshop reader, while the other articles can be found in the workshop proceedings [1].

[2] proposes a method for estimating the power consumption of components in the AADL (Architecture Analysis and Design Language) component assembly model, once deployed onto components in the AADL target platform model.

[3] proposes Visual Timed Scenarios (VTS) as a graphical property specification language for AADL. An effective translation from VTS to Time Petri Nets (TPN) has been devised, which enables model-checking of properties expressed in VTS over AADL models using TPN-based tools.

[4] describes a general methodology and an associated tool for translating AADL and its annex behavior specification into the BIP (Behavior Interaction Priority) language. This allows simulation of systems specified in AADL and application of formal verification techniques developed for BIP to these systems.

[5] discusses the transformation of a global requirements model into a system design. It describes an algorithm that derives the local behaviors for each system component, including coordination messages between the different system components.

[6] shows how higher-order model composition can be employed for constructing scalable models. An approach based on model transformation is presented, defining basic transformation rules operating on the graph structures of actor models.

[7] demonstrates an approach for early cross toolkit development based on the ISE ADL and the MetaDSP framework. It is designed to cope with hardware design changes, like changes in the instruction set of target CPUs.

[11] Discusses advantages and drawbacks of UML and SysML for modeling RF systems, based on a case study of a UMTS transceiver.

[8] propose the use of a new notion of modeling patterns for arbitrating between the real-time system designer's needs of expressive power and the restrictions that must be imposed so that models can be amenable to static analysis.

[9] proposes an approach based on UML-MARTE that allows system modelling with separation of concerns between the functional software model and the execution platform resources as well as timing constraints. Temporal characteristics and timing constraints can be specified at different abstraction levels, and scheduling analysis techniques can be applied on the resulted model.

[10] presents a model-based integration environment which uses a graphical architecture description language (EsMoL) to pull together control design, code and configuration generation, platform-specific resimulation, and a number of other features useful for taming the heterogeneity inherent in safety-critical embedded control system designs.

3 Discussion of Breakout Sessions

After the presentations, the participants broke into three different groups, each one focusing on a particular subject of interest. The following summarizes the conclusions of each breakout session.

Transition from requirements to architecture. There is a need for formalisation of requirements, better traceability, structuring and partitioning, validation, verification, and enforcement of requirements. Too often design decisions are already made during requirements decomposition. Given a requirements specification, the next step is to guide the developers towards a good architecture. A manual approach can be followed, by which the developer defines proper components, makes architectural choices, and allocates requirements to system elements. The result is analyzed and incrementallhy improved until a suitable system design has been found. Alternatively, a semi-automatic approach could be used by defining the constraints on the system, generating solutions that satisfy these constraints, and adding more constraints in order to reduce the solution space. The key difficulty for this approach is to find suitable constraints to add, in order to avoid inadvertently eliminating possible solutions. One could leverage patterns for obtaining detailed architectures and designs. The key issue here is how to deal with the interaction between patterns, since a system often has to deal with many competing concerns and accompanying patterns. Finally, after the system has been designed it has to be verified. A model refinement approach can insure correctness by construction, but a key concern is whether such can be done in an incremental manner, and how to resolve the different conflicting aspects. Another verification approach is by model analysis and model checking. The key question here is to build a relevant abstraction/analysis model in order to verify the properties of interest.

Design Languages. Modeling languages are in wide-spread use today, at least in the prototyping phase, but we are facing a scalability problem and many conflicting demands on a modeling language: Modeling languages are necessary for expressing our understanding of the problem. Modeling languages should also be used for existing software. They should provide ways to go back and forth between designs and existing code. They should support consistent views between different levels of abstraction. Modeling languages cannot not be generic, but need to be domain-oriented. However, while for some domains the mapping between the formal model and its implementation is well understood, for other domains this has yet to be demonstrated. The use of modeling languages relies essentially on tools which are heavily dependent on version changes, and users should be aware of that. Version control is a real problem. Education is very important, since when people enter the work force it is often too late to teach new methods. Teaching formal methods, or at least teaching rigorous modelling, is a must.

Transformations and platform (in)dependence. This discussion group concentrated on the meaning of platform independence and on the types of

transformations in which platform independent models can be involved. A model can only be independent from a specific understanding of what the platform is, and in order to achieve this independence it must include a model of the platform. Such a model could specify, for example, a specific model of computation, a set of services, a set of resources, or required QoS constraints. Only in the presence of such platform model can a model be qualified as platform independent, which means, independent with respect to a concrete platform which conforms to the constraints specified by the platform model. The platform model is often not explicit, which reduces the self-containment and the usability of models.

Acknowledgements. We thank all workshop participants for the lively discussions and the useful feedback we received. We thank the workshop program committee members for their helpful reviews. This workshop was supported by the ARTIST2 Network of Excellence on Embedded Systems Design (http://www.artist-embedded.org) and by the EUREKA-ITEA project SPICES (http://www.spices-itea.org).

References

1. Stefan Van Baelen, Iulian Ober, Susanne Graf, Mamoun Filali, Thomas Weigert, and Sébastien Gérard, editors. *ACES-MB 2008 Proceedings: First International Workshop on Model Based Architecting and Construction of Embedded Systems*. IRIT, Toulouse, France, 2008.
2. E. Senn, J. Laurent, and J.-P. Diguet. Multi-level power consumption modelling in the aadl design flow for dsp, gpp, and fpga. In Van Baelen et al. [1], pages 9–22.
3. D. Monteverde, A. Olivero, S. Yovine, and V. Braberman. Vts-based specification and verification of behavioral properties of aadl models. In Van Baelen et al. [1], pages 23–37.
4. M.Y. Chkouri, A. Robert, M. Bozga, and J. Sifakis. Translating aadl into bip - application to the verification of real-time systems. In Van Baelen et al. [1], pages 39–53.
5. G.v. Bochmann. Deriving component designs from global requirements. In Van Baelen et al. [1], pages 55–69.
6. T.H. Feng and E.A. Lee. Scalable models using model transformation. In Van Baelen et al. [1], pages 71–85.
7. N. Pakulin and V. Rubanov. Ise language: The adl for efficient development of cross toolkits. In Van Baelen et al. [1], pages 87–98.
8. M. Bordin, M. Panunzio, C. Santamaria, and T. Vardanega. A reinterpretation of patterns to increase the expressive power of model-driven engineering approaches. In Van Baelen et al. [1], pages 145–158.
9. M.-A. Peraldi-Frati and Y. Sorel. From high-level modelling of time in marte to real-time scheduling analysis. In Van Baelen et al. [1], pages 129–143.
10. J. Porter, G. Karsai, P. Völgyesi, H. Nine, P. Humke, G. Hemingway, R. Thibodeaux, and J. Sztipanovits. Towards model-based integration of tools and techniques for embedded control system design, verification, and implementation. In Van Baelen et al. [1], pages 99–113.
11. S. Lafi, R. Champagne, A.B. Kouki, and J. Belzile. Modeling radio-frequency front-ends using sysml: A case study of a umts transceiver. In Van Baelen et al. [1], pages 115–128.