REMES Tool-chain *

A Set of Integrated Tools for Behavioral Modeling and Analysis of Embedded Systems

Dinko Ivanov
Faculty of Mathematics and Informatics
University "St.Kliment Ohridski"
James Bourchier Blvd. 5, 1164 Sofia, Bulgaria
dinko.ivanov@gmail.com

Marin Orlić
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
marin.orlic@fer.hr

Cristina Seceleanu
Mälardalen Real-Time Research Centre
Mälardalen University
P.O. Box 883, 721 23 Västerås, Sweden
cristina.seceleanu@mdh.se

Aneta Vulgarakis
Mälardalen Real-Time Research Centre
Mälardalen University
P.O. Box 883, 721 23 Västerås, Sweden
aneta.vulgarakis@mdh.se

ABSTRACT

In this paper, we present a tool-chain for the REMES language, which can be used for the construction and analysis of embedded system behavioral models. The tool-chain consists of the following tools: (i) a REMES editor for modeling behaviors of embedded components, (ii) a REMES simulator to test timing and resource behavior prior to formal analysis, and (iii) an automated transformation from REMES to Priced Timed Automata, needed for formal analysis.

Categories and Subject Descriptors

I.6.4 [**Computing Methodologies**]: Simulation and Modeling—*Model Validation and Analysis*; D.2.2 [**Software Engineering**]: Design Tool and Techniques

General Terms

Design, Performance, Verification

Keywords

behavioral modeling, component-based software engineering, embedded systems, formal analysis, simulation

1. INTRODUCTION

The top challenge in embedded systems (ES) design is the construction of systems with predictable behavior. In contrast to desktop systems, the embedded behaviors encompass not only functionality but also behaviors generated by constrained resources (CPU, energy, or memory), or/and timing constraints. Hence, to ensure the ES predictable behavior, one must employ analysis techniques, such as simulation, or/and formal analysis of the system's model against various requirements.

As a first step towards achieving predictable systems, we have developed a hierarchical, timed behavioral language with graphical appeal, fit for a component-based ES design perspective. The crux of the language (see Figure 2), called the REsource Model for Embedded Systems (REMES) [6], is its ability to capture the resource-constrained and timing behavior of component-based ES, besides their functionality. REMES is inspired by CHARON [2], and, in comparison, treats resources as first-class modeling entities of discrete (e.g., memory) or continuous (e.g., energy) nature. REMES models can be transformed into the formal Timed Automata (TA) [3], or Priced Timed Automata (PTA) [1, 4], where one can compute the system's worst-case/optimal resource-usage, as well as trade-offs between possibly conflicting resource-driven requirements. This is carried out by model-checking the resulted formal models with UPPAAL[1], or/and UPPAAL CORA[2] tools.

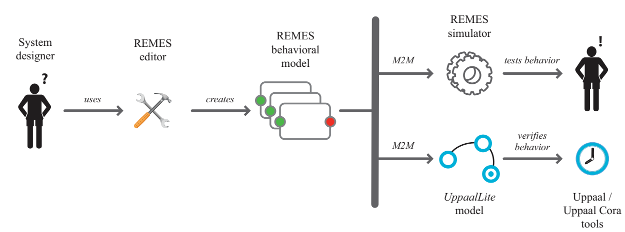

Figure 1: The REMES tool-chain workflow.

In this paper, we present a set of REMES tools that can be employed for the construction and analysis of ES behavioral models. Figure 1 illustrates the design flow implemented in our tool-chain. The designer uses: (i) a REMES *editor* (section 2) for building complex REMES models, (ii) a REMES *simulator* (section 3.1) that lets one test the timing and resource consumption of embedded

*This work was supported by SSF via the strategic research centre PROGRESS, the European Union under the ICT priority of the 7th Research Framework Programme in the context of the Q-ImPrESS research project, the Croatian Ministry of science, education and sports via the research project SOFTWARE ENGINEERING IN UBIQUITOUS COMPUTING, and the Unity Through Knowledge Fund via the DICES project. We would also like to thank Ivo Petkov for developing the first version of the REMES editor.

ASE'10, September 20–24, 2010, Antwerp, Belgium.
ACM 978-1-4503-0116-9/10/09.

[1]The UPPAAL tool is available at `http://uppaal.com/`.
[2]The UPPAAL CORA tool is available at `http://www.cs.aau.dk/~behrmann/cora/`.

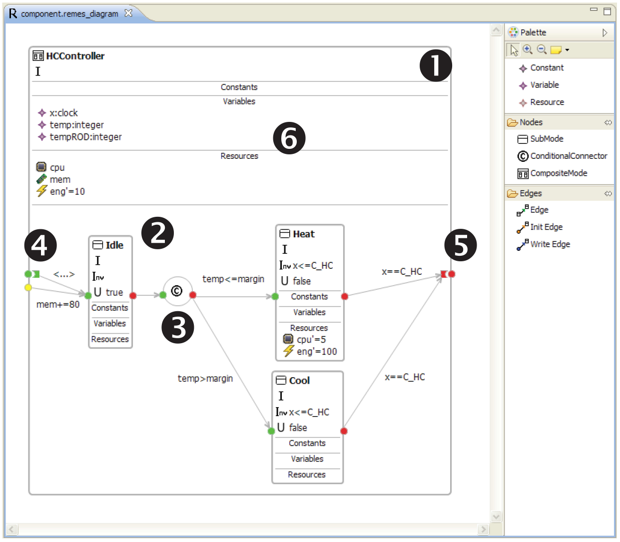

Figure 2: A screenshot of the REMES editor. A composite mode ❶ consists of several submodes (atomic modes) ❷ connected by edges and conditional connectors ❸. The modes are entered via their init- or entry-points ❹, and exited through their exit-points ❺. Each mode can have a number of associated constants, variables, and resources ❻, displayed in separate compartments.

components, and (iii) an automated *transformation* from REMES into PTA for formal analysis (section 3.2).

Giotto [5] is a related environment for ES design, which provides an abstract model for the implementation of embedded real-time control systems. Although Giotto uses timing and other platform-related model annotations, it does not consider the resource-wise system behavior, hence CPU, memory, energy constraints are not modeled and formally analyzed.

2. THE REMES EDITOR

The REMES editor[3] is a graphical environment for behavioral modeling embedded components, in the semantics of the REMES modeling language. The diagram editor is based on the Graphical Modeling Framework (GMF) and the Eclipse Modeling Framework (EMF).

The REMES editor allows the user to easily create REMES artifacts such as atomic and composite modes, edges or conditional connectors. Submodes and conditional connectors can be nested inside composite modes. The user defines the control flow by creating edges between diagram elements. The action guards of the edges are defined by in-place editing in the diagram. Basic checks are performed to prevent the user from creating invalid models. The user can define typed variables, constants and resources in separate sections within the modes. The REMES diagram editor integrates with the Eclipse properties view, which displays and edits context sensitive information for the currently selected diagram element. Filters can be applied over the REMES diagram to outline a particular aspect of the REMES model – behavior, timing or resource usage. Figure 2 shows a screenshot of the REMES editor displaying some behavioral model. The editor can be distributed separately or integrated with an architectural modeler (e.g., PRIDE[4]).

3. SIMULATION AND FORMAL ANALYSIS OF REMES MODELS

3.1 The REMES simulator

Simulating and testing the system behaviors as they are being designed can provide valuable input to the designer. Once completed, REMES models can be model-checked in tools like UPPAAL. We use model-to-text transformations (M2T) to transform behaviors into source code that simulates the modeled system. The transformation is performed over intermediate models created from REMES diagrams that retain the internal behavior structure. The user can then run the simulator which updates mode variables and resources in each simulation round, based on passed time. The system designer can visualize the mode transitions, the clock- and variable changes in the simulator output.

3.2 Transforming REMES into PTA

The transformation of REMES models into PTA is implemented by the model-to-model (M2M) transformation language ATL (Atlas Transformation Language). The basic transformation rules are as follows: (i) a REMES diagram (see Figure 2) is transformed to a network of PTA, (ii) a composite mode ❶ is transformed to a single PTA, (iii) submodes ❷, init-, entry-, and exit points, ❹ ❺, are transformed to PTA locations, (iv) edges are copied to PTA edges, (v) conditional connectors ❸ are removed in PTA, (vi) invariants and guards referring to resources are translated to a PTA cost variable, as a weighted sum of REMES resources (declared in ❻), and (vii) an additional start location is added to each PTA to allow initialization.

The transformation rules applied to REMES diagrams result in *UppaalLite* models representing the same behavior. A graphical editor for *UppaalLite* models is provided, as a tool to visually inspect transformation results. Model files for both UPPAAL (TA) and UPPAAL CORA (PTA) can be exported for formal verification and analysis.

When using the REMES tool-chain within integrated development environments, e.g., PRIDE, the transformation uses component triggering information from architectural models, to insert appropriate synchronization channels into the resulting PTA.

4. REFERENCES

[1] R. Alur. Optimal Paths in Weighted Timed Automata. In *Proceedings of HSCC'01: Hybrid Systems: Computation and Control*, pages 49–62. Springer, 2001.

[2] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivančić, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky. Hierarchical Modeling and Analysis of Embedded Systems. In *Proceedings of the IEEE*, 8(3):231–274, 1987.

[3] R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[4] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-Cost Reachability for Priced Timed Automata. In *Proceedings of HSCC'01*, number 2034 in LNCS, pp. 147–161. Springer–Verlag, 2001.

[5] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: A Time-Triggered Language for Embedded Programming. In *Proceedings of the IEEE*, pp. 166–184. Springer-Verlag, 2000.

[6] C. Seceleanu, A. Vulgarakis, and P. Pettersson. REMES: A Resource Model for Embedded Systems. In *Proceedings of ICECCS 2009*. IEEE Computer Society, June 2009.

[3]The REMES tool-chain is available at `http://www.fer.hr/dices/remes-ide`.

[4]The PRIDE toolset is available at `http://www.idt.mdh.se/pride/`.