Model-based development of embedded systems - the MADES approach (Extended Abstract)

Neil C. Audsley, Ian Gray*, Leandro Soares Indrusiak,
Dimitris Kolovos, Nikos Matragkas*, Richard Paige
University of York, York, U.K.

***Index Terms*—Model Driven Engineering, Epsilon, MADES, Embedded Systems, Real-Time Systems.**

I. Introduction

The architectures of embedded systems are becoming increasingly non-standard and application-specific. They frequently contain multiple heterogenous processing cores, non-uniform memory, complex interconnect or custom hardware elements such as DSP and SIMD cores. However, programming languages have traditionally assumed a single processor architecture with a uniform logical address space and have abstracted away from hardware implementation details. As a result, developing software for these architectures can be challenging. Equally, such systems are frequently deployed in high-integrity or safety-critical systems which require the highest levels of predictability and reliability.

The MADES Project is an EU-funded project that aims to use model-driven techniques to enable the development of the next generation of highly complex embedded systems, whilst reducing development costs and increasing reliability and predictability. In this paper we provide an overview of the MADES approach to model-driven development of embedded systems.

II. MADES Project Goals

The MADES project aims to develop the elements of a fully model-driven approach for the design, validation, simulation, and code generation of complex embedded systems to improve the current practice in the field. MADES differentiates itself from similar projects in that way that it covers all the phases of the development process: from system specification and design down to code generation, validation and deployment. Design activities exploit a dedicated language developed on top of the OMG standard MARTE (Modeling and Analysis of Real-time and Embedded systems) [2], and foster the reuse of components by annotating them with properties and constraints to aid selection and enforce overall consistency.

Validation activities comprise the verification of key properties of designed artifacts and of the transformations used throughout the development process, and also the closed-loop simulation of the entire system. Code generation addresses both conventional programming languages (e.g., C) and hardware description languages (e.g., VHDL), and uses the novel technique of Compile-Time Virtualisation to smooth the impact of the diverse elements of modern hardware architectures and cope with their increasing complexity.

*Supported by EU Framework 7 research contract FP7-248864

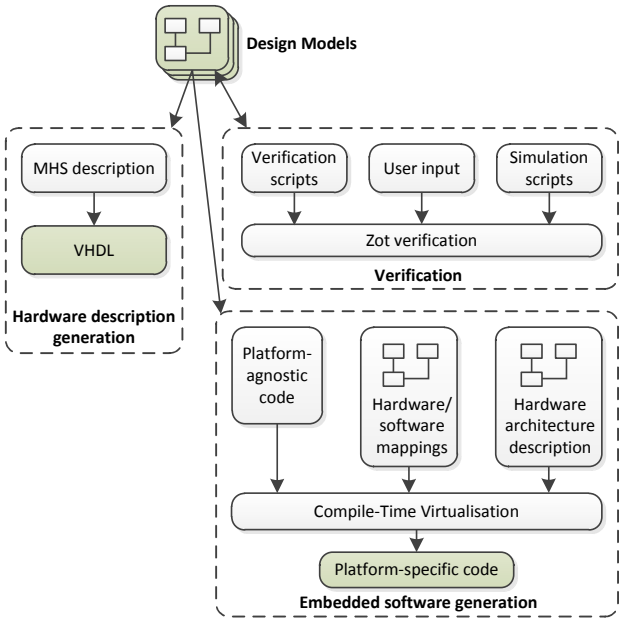

Fig. 1. Overview of the artifacts in the MADES approach

All these aspects will be fully supported by prototype tools integrated in a single framework, and will be thoroughly validated on real-life case studies in the surveillance and avionic domains. The project also aims to develop a handbook to provide detailed guidelines on how to use MADES tools in the development of embedded systems and promote their adoption.

III. The MADES Approach

A conceptual model of the inter-relationships between the various artifacts of the MADES approach is illustrated in figure 1.

One of the unique characteristics of the MADES approach is the use of model-driven transformations which are used to transform one or more input specifications into one or more output specifications. Model transformation languages are defined in a metamodel level and establish the relationship between source metamodel elements and target metamodel elements (see figure 2).

The MADES approach focusses on three areas:

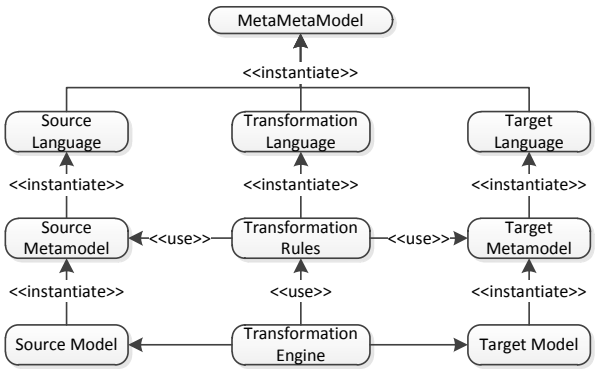

Fig. 2. MADES Mapping Scheme

- Generation of platform-specific embedded software from architecturally-neutral software specifications.
- Generation of hardware descriptions of the modelled target architecture.
- Verification of functional and non-functional properties.

Development effort starts with building design and analysis models using the MADES modelling language, which is based on a combined subset of OMG MARTE [2] (a UML profile for modelling real-time embedded systems) and SysML [4] (a general-purpose modelling language for systems engineering applications). The language aims to overcome shortcomings of these existing languages and to provide:

- Specifications of functionality in an architecturally-neutral way.
- Descriptions of target hardware.
- Deployment diagrams that map functionality to hardware/software.
- Timing and non-functional properties for early and frequent verification.
- Code-reuse, component-based design, and maintainability.

Verification and simulation play a key role in the MADES approach. Such activities are present during:

- Verification of key properties on designed artifacts (for example, whether a system will meet a specified deadline, or be able to support a specified volume of data).
- Closed-loop simulation based on detailed models of the environment (for functional testing and early validation).
- Verification of designed transformations (from high-level system models down to low-level hardware/software implementations).

In order to provide verification and simulation in the MADES toolset, the Zot tool [3] is used. The verification phase aims to provide rapid and early verification of the system to reduce design time and guarantee correctness of the final system.

The MADES code generation phase allows the designer to model the target hardware at a high-level of abstraction and use deployment diagrams to map the input code (which is provided in an architecturally-neutral form) to elements of the target hardware. A technique called Compile-Time Virtualisation (CTV) is used to hide the complexity of embedded software development through the provision of a Virtual Platform (VP). The VP is an idealised view of the underlying hardware which provides a simple programming model that is compatible with the chosen source langauge. The result of this is that the programmer can write code as if it is to be executed on the VP, and CTV will then automatically retarget this code for execution on the actual platform. If the actual platform changes during development (for example, due to hardware redesign or changing requirements) the same input code can be automatically retargeted to the new hardware and does not manually ported.

Finally, the MADES approach also considers generation of synthesisable hardware descriptions from the hardware model. The deployment mappings of the code generation phase use a high-level description of the capabilities of the desired target architecture (for example, “three processors connected with a common bus, two banks of shared memory”). This can be reified into an unambiguous hardware description for implementation using the MADES approach.

The Epsilon platform [1] is used to implement both the model-to-model and model-to-text transformations used in the MADES approach. Epsilon (Extensible Platform of Integrated Languages for mOdel maNagement) is a platform for building consistent and inter-operable task-specific languages for model management tasks such as model transformation, code generation, model comparison, merging, refactoring and validation. Epsilon can provide traceability information produced by the various transformations, which is of paramount importance for embedded systems design due to the need to comply to particular standards such as the DO-178B Standard.

IV. Conclusion

The MADES project aims to use model-driven engineering techniques to aid the development of embedded systems. It uses a systems modelling language based on MARTE and SysML that allows the developer to express their system at a high-level of abstraction, and then to iteratively refine their design to reach the final implementation. MADES differentiates itself from similar work through three unique features. First, extensive use of model transformations is used to facilitate development and provide traceability. Second, verification and validation are key parts of the MADES design flow, allowing early and frequent verification of the system being developed. Third, Compile-Time Virtualisation (CTV) is used to assist the development of embedded software.

References

[1] D. S. Kolovos, R. F. Paige, and F. A. C. Polack. Eclipse development tools for epsilon. In *In Eclipse Summit Europe, Eclipse Modeling Symposium*, 2006.

[2] Object Management Group. UML profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. http://www.omgmarte.org/, November 2009.

[3] M. Pradella, A. Morzenti, and P. San Pietro. The symmetry of the past and of the future: bi-infinite time in the verification of temporal properties. In *ESEC-FSE ’07*, pages 312–320, New York, NY, USA, 2007. ACM.

[4] T. Weilkiens. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.