System to software level model refinement in embedded system design

Iulia Dragomir, Iulian Ober, and Christian Percebois

Université de Toulouse - IRIT
118 Route de Narbonne, 31062 Toulouse, France
{iulia.dragomir,iulian.ober,christian.percebois}@irit.fr

Embedded systems refer to systems integrating both hardware and software, which function in a particular applicative environment and are responsible for executing functions like control, monitoring, etc. Nowadays, embedded systems are involved in various application areas like automotive, energy grid, consumer electronics, aeronautics and space vehicles. Usually real-time and communicating systems, they are often critical for the environment's functioning as a whole. A correct and safe development method is needed in order to ensure their critical properties.

The development process for an embedded system, such as the ASSERT process [3] on which our work is based, is, typically, a process in V. This process consists in two branches: (1) The descending branch: starting from a coarse grained architecture derived directly from the requirements of the system, the design and coding phases proceed by iterative decompositions and refinements. (2) The ascending branch: the components are integrated with the use of specific validations at each step.

Assuming that the models used in the design and coding iterations (descending branch of the V) have a formal semantics, a refinement relation between such models, representing the fact that a more detailed model conforms to a more abstract one, can be defined as a binary relation on the semantic domain. In order for such a relation to support *correct* and *safe* development, two conditions have to be met: (1) A (semi-)automatic method for proving that the relation holds on two given models has to be provided. (2) The relation has to preserve the desired properties of the system (e.g., safety properties), so that if an abstract model verifies such a property, the refined model preserves that property.

Formal refinement relations between specifications and implementations are a subject that has been extensively studied. These relationships have been studied in the domain of algebraic specification (e.g., [1, 2]), in automata-theoretic models (beginning with [7]) or, more recently, for synchronous programs [6].

The existing results are, in general, based on the specifications and implementations expressed in the same semantic framework. However, in the development of embedded systems, the two models involved in the refinement relation often rely on different formalisms. For example, high-level system specifications are modelled in an asynchronous framework, while implementations are often realized in a synchronous language. The lack of a unitary framework and theory, allowing the users to model the system at different levels of refinement, is a major problem in practice. For example, one can express the detailed architecture of a

system in AADL [11], including the hardware and software components, connections between them, data streams, etc., but the language is limited to capture the functional aspects of the components. The SysML language [9] allows capturing these functional aspects by block diagrams, state machine diagrams, etc., but does not provide support for the connection with the architecture described in AADL. Finally, when such architecture needs to be refined in order to get the code for hardware and software components in VHDL, SystemC, SCADE or any other appropriate implementation language, it is very difficult to automate or validate the refinement.

Our work focuses on defining a unitary theoretical framework that allows us to formalize and study the relationships between asynchronous specifications and synchronous implementations. We plan to do this by defining a *hybrid* semantic model, capable of encompassing both asynchronous and synchronous elements. A refinement relation has then to be defined on this semantic model, so that, starting from a completely asynchronous high-level model, one can reach a synchronous implementation in a given number of verifiable refinement steps.

In the early stages of our research, we have developed an extension ([8]) of the OMEGA UML toolset [5] for capturing the new features of UML 2.x [10] and partially SysML and verified it with realistic models. Currently we are working on the formalization of hybrid models in SysML, using annotations and defining a unified semantic model. The next step consists in the definition of the temporal and functional refinement relation, the method for its computation and the proofs of its properties.

This study is the first to our knowledge which attempts to converge asynchronous specifications based on automata with synchronous dataflow-oriented implementations.

References

[1] Abrial, J.R. The B-Book. Cambridge University Press, Cambridge. 1995.

[2] Börger, E. The ASM Refinement Method. Formal Aspects of Computing, vol. 15, 237-257. 2003.

[3] Conquet, E. ASSERT: a step towards reliable and scientific system and software engineering. 4th European Congress on Embedded Real Time Software, Toulouse, France. 2008.

[4] Halbwachs, N., Caspi, P., Raymond, P., and Pilaud, D. The synchronous dataflow programming language Lustre. Proc. IEEE 79(9), 1305-1320. 1991.

[5] OMEGA UML Toolset. Via http://www-omega.imag.fr/tools/IFx/IFx.php.

[6] Mikác, J., and Caspi, P. Flush: an example of development by refinements in SCADE/Lustre. STTT, Vol. 11, No. 5, pp. 409-418. Springer, 2009.

[7] Milner, R. An algebraic definition of simulation between programs. 2nd International Joint Conferences on Artificial Intelligence, London, 1971

[8] Ober, I., and Dragomir, I. OMEGA2: A new version of the profile and the tools (regular paper). In *UML & AADL'2009 - 14th IEEE ICECCS*, pages 373–378. IEEE, 2010.

[9] Object Management Group. Systems Modeling Language, v1.1. Via http://www.omg.org/spec/SysML/1.1

[10] ObjectManagement Group. Unified Modeling Language, v2.2. Via http://www.omg.org/spec/UML/2.2.
[11] SAE. Architecture analysis and design language (AADL). Document No. AS5506/1, 2004. Via http://www.sae.org/technical/standards/AS5506/1.