Chapter 14
UML and SystemC – A Comparison and Mapping Rules for Automatic Code Generation

Per Andersson and Martin Höst

Abstract  Today embedded system development is a complex task. To aid the engineers new methodologies and languages are emerging. During the development the system is modeled using different tools and languages. Transformations between the models are traditionally done manually. We investigate the automation of this process, specifically we are looking at automatic UML to SystemC transformation. In this paper we compare UML and SystemC, focusing on communication modeling. We also present mapping rules for automatic SystemC code generation from UML. The mapping has been implemented in our UML to SystemC code generator.

Keywords  code generation, UML, systemC

14.1 Introduction

Today there is a never ending demand for new functionality to be included in embedded systems such as mobile phones. This leads to increased design complexity. To overcome the increased system complexity new design methodologies, such as model driven architecture, have been introduced. In parallel with this, new languages, i.e. SystemC [2, 3], for system level modeling and simulation have also emerged. Combining new methodologies and new languages is a promising approach to manage the increasing system complexity. This is the focus of the MARTES (Model-Based Approach for Real-Time Embedded Systems development) project. In the project we investigate how UML and SystemC can be used together when the ideas of Model Driven Architecture are applied. One of the tasks of the project is to investigate how transformations from UML to SystemC can be

Lund University, P.O. Box 118, SE-221 00 Lund, Sweden
Email: Per.Andersson@cs.lth.se and Martin.Host@cs.lth.se

1www.MARTES-ITEA.org
automated and supported by tools. During this research we are developing a prototype tool, which manage the UML to SystemC transformations and code generation, as an add-in to the Telelogic TAU UML2 modeling tool. This part of the MARTES project is carried out in close cooperation between Lund University and Telelogic. In this research there are a number of decisions that needs to be taken related to the detailed requirements on the developed tool. It is crucial to take the right decisions concerning what functionality to include in the tool. This is achieved by developing the tool iteratively. Different versions are developed after each other, and every version is evaluated in order to decide what additional functionality to include in the next version. Evaluations are a very important part of the development of the tool. The evaluations are being carried out together with other partners in the MARTES project, in the context of case studies in industrial projects. In this paper we present the work and results from developing the first version of our UML to SystemC code generator. We start with a summary of related work in Section 14.2. We compare the constructs and semantics of UML and SystemC in Section 14.3. Based on this comparison we have developed a set of mapping rules which are detailed and motivated in Section 14.4. Our implementation of the mapping rules is presented in Section 14.5 and practical experience can be found in Section 14.6. Finally the paper is concluded in Section 14.7.

14.2 Related Work

Earlier publications on UML to SystemC mapping [4, 7] suggest that, to a large extent, there is a one to one relation between concepts in the two languages. For example a UML class is mapped to a SystemC module. This is not always desirable, sometimes UML classes are used for data encapsulation and in these cases they should remain as classes in the SystemC model. Only UML classes with ports, and/or with architecture should be mapped to SystemC modules.

Riccobene et al. [7] address this by exposing all SystemC details in the UML model through a SystemC profile. Their approach is to use UML as an implementation language for SystemC. In addition to making all standard SystemC types available at the UML level they also extend actions in state machines to handle SC_THREAD and SC_METHOD synchronization. With their approach, the engineers must tag their UML model by adding relations to the intended SystemC elements. This is similar to the last part in our design process. In our design process engineers start with an abstract UML model, which is refined in three steps. This is further explained in Section 14.5. One difference compared to their work is that we intend to automate most of this part in our process, minimizing the design effort. Another problem with bringing too many of the SystemC details into the UML model is the semantic differences between the languages, as discussed in Section

2www.telelogic.com