

UML to SystemC Transformation in the MARTES Project

Per Andersson
Lund University
P.O. Box 118
SE-221 00 Lund
Sweden
Per.Andersson@cs.lth.se

Martin Höst
Lund University
P.O. Box 118
SE-221 00 Lund
Sweden
Martin.Host@telecom.lth.se

Mats Bergström
Telelogic
PO Box 4128, Kungsgatan 6
SE-203 12 Malmö
Sweden
Mats.Bergstrom@telelogic.com

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. New languages, i.e. SystemC, for system level modelling 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) project¹. 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 automated and supported by tools. During this research a prototype tool, which manage the UML to SystemC transformations and code generation, is under development as an ad-in to the Telelogic-TAU UML2 modelling tool². This part of the MARTES project is done in close cooperation between Lund University and Telelogic.

In this project there is a number of decisions that needs to be taken related to the detailed requirements on the developed tool, not at least because the available resources in the project are limited. This means it is crucial the right decisions concerning what functionality to include in the tool are taken. This is achieved by developing the tool iteratively within the MARTES project. This means that 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. That is, 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 a case study on multimedia services in mobile terminals. The purpose of the case study is to perform behavioural simulation and performance estimations on the developed system.

2 Language Differences

Earlier publications on UML to SystemC mapping [1] [2] suggest that, to a large extent, there is a one to one relation between concepts in the 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. [2] handle 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` synchronisation. 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, step 3, in our approach described below, but we intend to automate most of this process minimising the design effort.

Another concept which differs between the languages is channels. In SystemC channels are more elaborated compared to UML. In UML a channel is a passive object, it has a name and possibly a type. There is limited capability to give a channel behaviour or state. In SystemC hierarchical channels can have both active processes and state, i.e. attributes. To model such a channel in UML it is natural to use active classes. This adds complexity to the UML to SystemC mapping. An active UML class should be mapped either to a SystemC `sc_module`, or a hierarchical channel, depending on its behaviour, i.e. computation/control or communication. Due to space constraint it is not possible to

¹www.martes-itea.org

²www.telelogic.com

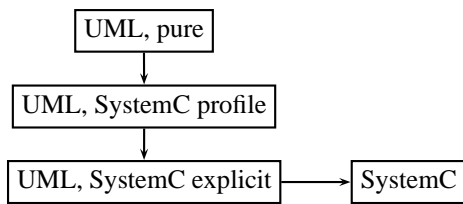


Figure 1. The pure UML model is mapped to SystemC code in three steps. This Allows the engineer to influence the details of the mapping.

give a complete list of differences between UML and SystemC, or a complete list of our mapping rules in this paper.

3 Mapping Process

During initial system modelling a pure UML model is used. Though it is possible to define a set of mapping rules from a pure UML model directly into SystemC code, it would give the engineer little influence of the mapping and most likely a less satisfactory result. Instead we divide the mapping into three steps, as depicted in figure 3. All models are available and editable. This makes it possible for the engineer to have full control over the relevant details for the system under development and have the tool manage all remaining details.

Step 1, vertical refinement transformation: In this step an initial UML description is refined to a UML description, which follows a UML profile for SystemC. This step will be partially done manually will, at least partly, be carried out manually. There will be a set of default values for the SystemC specific attributes of the UML profile, which in most cases will provide satisfactory mapping in the following transformation steps.

Step 2, vertical refinement transformation: In this step the model is transformed into a new UML description that only includes UML constructs with direct representations in SystemC, viz. classes, attributes, inheritance etc. Other constructs such as state machines are translated to the target language. During this step we transform each state machines to a class with methods that implement the behaviour of the states and transactions. In the first version of the tool the resulting model will be an un-timed functional model. A complete list of UML constructs which are removed during this transformation it is beyond the scope of this paper.

In addition to removing UML only concepts we also make all relations in the model explicit. When a class is made active in UML it implies that the class will have its own thread of execution. In SystemC this is realised using

`sc_thread` or `sc_method` which implies that the class is an instance of the SystemC class `sc_module`. During this transformation all such implicit relations are made explicit. For example, we add a generalisation relation to the SystemC class `sc_module` to all active UML classes. In the first version of the tool the resulting model will be an un-timed functional model.

Step 3, horizontal transformation: In this step the UML model resulting from step 2 is transformed into a corresponding SystemC code. This transformation is a one to one correspondence between the UML model to the resulting SystemC code, i.e. this is a pretty print of the UML model. This step is implemented using the existing C++ code generator from Telelogic and thus reuse its support for scope rules, header-file inclusion and make file generation without any modifications. If the generated code is to be read by humans it is desirable to use the common SystemC macros when applicable. This requires a slight customisation of the syntax of the generated code. The next release of the code generator, planned to be released during the summer 2006, this will be possible, i.e. generate SystemC like module definitions, instead of a C++ class definition (`SC_MODULE(MyModule){...}` instead of `class MyModule:public sc_module{...}`).

The three steps together form an oblique transformation. This oblique transformation is the focus of the work of Lund University in the MARTES project. However, in practise the discussed transformations are followed by continued transformations during product development.

4 Conclusion

Combining new methodologies and new languages is a promising technique to overcome the increased complexity of today's embedded systems. This is the driving force in the MARTES project. The focus of this paper is UML to SystemC transformation and code generation. We discuss differences between the languages. We also present our transformation technique which makes it possible to reuse large parts of a code generator for similar target languages, i.e SystemC and C++. The technique also makes it possible to easily adapt the generated code for different run-time environments and code styles.

References

- [1] D. Kathy, Z. S. Nguyen, and T. P.S. System driven SoC Design Via Executable UML to SystemC. *Real-Time Systems Symposium*, 2004.
- [2] E. Riccobene, P. Scandurra, and S. Rosti, A. Bocchio. A SoC Design Methodology Involving a UML 2.0 Profile for SystemC. *Design Automation and Test Europe (DATE)*, 2005.