Supporting Model-Based Simulation of Embedded Systems by Coupling Tools

Jozef Hooman[1], Nataliya Mulyar[2], Ladislau Posta[2]

[1]Embedded Systems Institute, Eindhoven & University of Nijmegen, the Netherlands
[2]Eindhoven University of Technology, the Netherlands

Abstract—**The aim of this work is to support the multi-disciplinary development of real-time embedded systems by combining tools of different disciplines. As a concrete example, we have coupled a UML-based CASE tool (Rose RealTime) and Simulink to allow simultaneous simulation. Since Rose RealTime does not have a well-defined notion of timed simulation, we have used the simulation time of Simulink also for Rose RealTime. An intermediate component has been implemented, which realizes time synchronization and data exchange between the tools.**

Keywords—**Embedded systems; model-based development; simulation; real-time; UML; Simulink**

I. INTRODUCTION

This research has been carried out in the context of the Boderc* project. The main aim of this project is to improve multi-disciplinary system design, that is, combining mechanics, electronics and informatics, such that e.g. system-level decisions can be analyzed and consequences of design decisions can be predicted as early as possible. To achieve this goal, strong emphasis is put on high-level models and the combination of models from different disciplines.

There are several formal approaches that allow checking properties of hybrid systems (modeling both discrete and continuous aspects), such as HyTech [4] and Checkmate [2]. Exhaustive checking, however, is limited to relatively small models, and it often requires separate re-modeling and abstraction, which are rather time-consuming. Hence, there is a need for verification and validation methods that can be applied to the models the engineers are used to work with, such as Matlab/Simulink models and, for instance, software models represented in the Unified Modeling Language (UML) [1]. In this paper, we aim at the simultaneous simulation of the (often quite large) domain models that are used by each discipline, e.g. simulating a UML model of a software engineer in combination with a mechanical model.

Since tools are very important in the everyday practice of engineers, we have investigated the possibilities for coupling concrete modeling tools, namely Mathworks' Matlab/Simulink (which is used heavily in many areas, including transportation) and a UML-based CASE tool for real-time systems. Because of pragmatic reasons (industrial contacts, licenses), we have chosen the UML-tool Rose RealTime (Rose-RT for short) of IBM/Rational. This tool supports the ROOM methodology [5] for the development of software for real-time reactive systems. Note that Rose-RT allows code generation for particular target platforms, so there is a direct connection between a model and generated code. However, we could have chosen other UML-tools in the embedded domain, such as Telelogic Tau, Rhapsody of I-Logix, or Real-Time Studio of Artisan.

Our aim is to establish a tool coupling, which allows the combination of a model of a physical dynamical system in Simulink, e.g. representing one or more motors, with a discrete control algorithm in Rose-RT, as depicted in Figure 1.

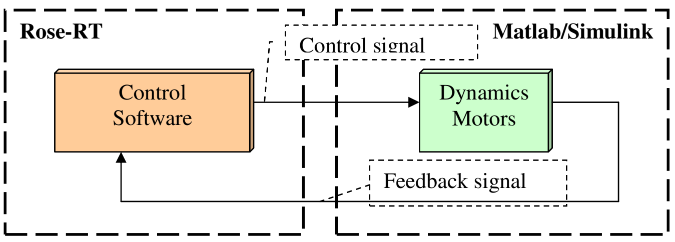

Figure 1 Combined Models

* This work has been carried out as part of the Boderc project under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Senter TS program.

By establishing a proper notion of simultaneous simulation of these models, one can quickly investigate the effect of changes in the control strategy, software execution times, or concerning the characteristics of the motors. Also note that this allows a comparison with a model where part of the control (e.g. low-level motor control) is modeled in the Matlab environment (e.g. using TrueTime [6]) and a supervisory control part in Rose-RT.

Realizing the desired tool coupling is far from trivial. The two main challenges are:

- Conceptual correctness. The coupling should be such that the simultaneous simulation of models in both tools should give meaningful results. In particular, this means that there should be a common notion of time in combination with a proper exchange of data and messages.
- Technical implementation. Suitable ways should be found to allow the tools to communicate and to run a simulation mode simultaneously. Moreover, the coupling software should be properly designed to allow, for instance, a change to another UML tool.

Note that also TrueTime could have been used to represent discrete real-time control, but we have chosen a UML-based CASE tool, since UML is becoming an ad hoc standard in software engineering and it allows a coupling with large object-oriented software architectures which includes not only control but also other aspects such as error handling and diagnosis. Moreover, such a tool enables code generation for different platforms.

As far as we know, such a coupling between Rose-RT (or similar real-time UML tools) and Simulink models has not been realized before. Related is the work on the High Level Architecture (HLA) [3], a general-purpose architecture for the coupling of simulation tools. However, HLA could not be used for our purpose, because UML-case tools, such as Rational Rose RealTime do not fit into this framework; they do not have the required simulation mode with a well-defined notion of simulation time.

In the rest of this paper, we very briefly present the tools used (Rose-RT and Simulink), in Sections II and III respectively, describing them only as far as needed to understand the coupling. The main concepts of the coupling are described in Section IV. Concluding remarks can be found in Section V.

II. ROSE REALTIME

Rose-RT is a tool for the object-oriented development of complex reactive software. It uses the visual modeling language UML to express software designs and supports model-driven development by allowing executable code generation from the UML models for a particular target platform on which the model is intended to run. In general, a Rose-RT model is automatically linked with a services library specific to this particular target platform. For instance, a model may use timers to express real-time behavior and the precision of the timers depends on the granularity of the timing service provided by the underlying platform.

Typically, a UML model in Rose-RT consists of a number of concurrent active objects (also called capsules), which communicate by sending and receiving messages via ports. The behavior of a capsule is modeled by means of a state diagram, a hierarchical state machine. Transitions in a state diagram are triggered by the receipt of messages or time-outs. Actions on a transition may change local variables, send messages, or set timers.

For each capsule, the incoming messages are queued and processed (according to their priority) in a run-to-completion manner, that is, the response to a message is computed completely, without interruption, before the next message is considered.

The Rose-RT tool can generate code for a particular target platform and then the model can be executed on this platform. Alternatively, it allows a user-controlled step-by-step execution, but then correct timing is not guaranteed and only the reactive response to messages can be tested.

III. MATLAB/SIMULINK

Simulink of MathWorks is a widely used tool for modeling and simulating dynamical systems. A Simulink model is represented graphically by means of a number of interconnected blocks. A block may produce output continuously or only at specific points in time, lines between blocks connect block outputs to block inputs. Blocks may have states, which may consist of a discrete and a continuous part.

The output of a block is computed by a function, based on its input and its current state and time. Similarly, an update function calculates the next discrete state. A derivative function relates the derivatives of the continuous part of the state to time and the current values of the inputs and the state.

Running a simulation of a Simulink model means that outputs, inputs and states are computed at certain intervals, from the simulation start time to the simulation end time. All time-related tasks are performed by a so-called solver, a Simulink-specific program. In fact, there are several types of solvers, depending on, e.g., whether the step size is fixed or variable, and the integration method used.

IV. MAIN CONCEPTS OF THE COUPLING

In this section we describe the main decisions taken to establish a correct coupling and the conceptual architecture, which realizes this coupling.

The most important decision concerns the notion of time to be used for the simulation. First we observed that the timing of Rose-RT is strongly coupled to the timing service of the operating system of the target system on which the model is running. Moreover, timing is not respected in the step-by-step simulation. Hence, we concluded that the timing of Rose-RT is not suitable for our purpose and decided to use the notion of simulated time of Simulink instead. The alternative is to use a separate, independent, notion of time, but this would also require new implementations of solvers, redoing a lot of things already available in Simulink.
To be able to establish a proper global notion of time, which faithfully reflects the execution of both models, somehow the execution time of the transitions in the Rose-RT model has to be taken into account. We assume that this information is available, representing an assumption on the underlying platform.

Another decision to be taken is the global architecture of the coupling. A possible approach, depicted in Figure 2, is to extend each tool with a specific component, which communicates directly with the other tool.

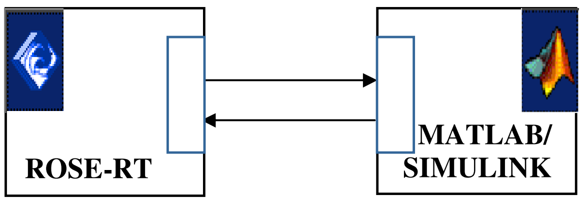

Figure 2 Tightly Coupled Tools

Instead of such a tight coupling, we decided to use a more loosely coupled architecture by introducing a third component called Multidisciplinary Coupling Tool (MCT), as shown in Figure 3. Observe that each tool contains an add-in, which is responsible for the communication with the MCT component.

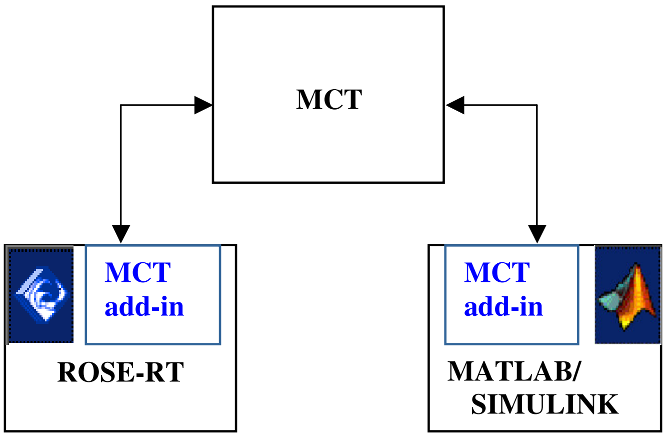

Figure 3 Loosely Coupled Architecture

By introducing such an MCT interface the modeling tools do not need to know about each other and it becomes much easier to change, for instance, switching to another UML-based CASE tool.

To obtain proper timing of the UML models, we have redefined the timing service of Rose-RT such that it gets the current notion of time from the MCT component. Moreover, the UML model has to be extended by specifying the assumptions on the execution times of all transitions. The Simulink model is extended with blocks that obtain data and timing info from the Rose-RT model via the MCT and ensure that the data is used at the appropriate moment of time. Also the setting of a timer in the UML model is communicated to these added Simulink blocks, which then generate the time-out.

To run a simulation, we have to initialize the simulation environment as follows:

1. Start Rose-RT and open a UML model which has been prepared for coupling
2. Start Matlab/Simulink and open a Simulink model which has been prepared for coupling
3. Start the simulation in the Simulink model, which:
    a. Starts the Simulink timer
    b. Sets the time of MCT to the current Simulink simulation time
    c. Starts the Rose-RT Target Run-Time System; the Rose-RT environment then requests the system clock, which by our modifications means that it gets the time of the MCT, next the Rose-RT model waits for external triggers

Next we describe an example of typical steps performed during a part of the simulation, depicted in the sequence diagram of Figure 4.

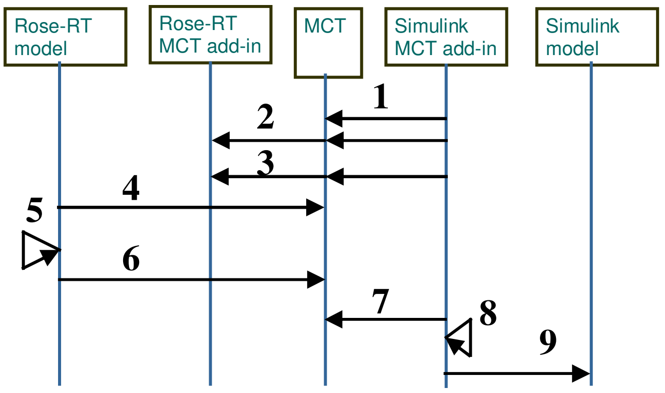

Figure 4 Interactions During Simulation

The numbered interactions represent the following actions:

1. The *Simulink MCT add-in* sets the time of the MCT to the current Simulink time. The Simulink model is frozen from this moment.
2. The *Simulink MCT add-in* sends an external event to the Rose-RT model; there this message is put to the queue.
3. The *Simulink MCT add-in* gives a command to perform one step in Rose-RT.
4. Rose-RT updates its clock with the value taken from the MCT.
5. Rose-RT processes a message from the queue; and executes a transition triggered by this message (if any).
6. Rose-RT sends the calculated data together with the transition duration to the MCT.
7. The *Simulink MCT add-in* reads the data and the execution delay set by Rose-RT in the MCT
8. Simulink ensures that Simulink simulation time advances with the value of the received delay. The Simulink model is resumed from this moment.
9. The *Simulink MCT add-in* passes the data to the Simulink model and the simulation loop continues.

Figure 5 show a more detailed architectural view of the implementation, showing for instance in more detail how the timing of a UML model is obtained from the MCT.

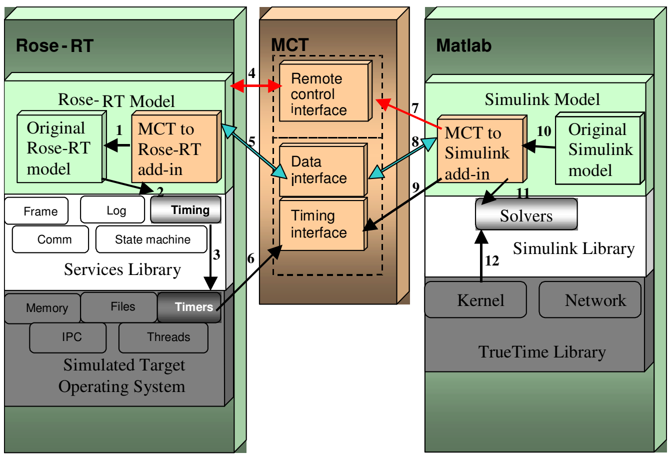

Figure 5 Module Architecture View

V. Concluding Remarks

The current version of the coupling has been applied to a few small examples, showing the feasibility and usefulness of such a simultaneous simulation on a small scale. Clearly, much more experiments with larger industrial examples are needed to investigate the performance for complex systems. Future work also includes the removal of a few simplifications that have been made to obtain a first prototype quickly. For instance, at the moment only one timer is allowed in the UML model. Currently, the models are simulated on a single PC, but we also intend to investigate a distributed implementation, which couples models on different PCs.

References

[1] G. Booch, J. Rumbaugh, I. Jacobson *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[2] E.M. Clarke, A Fehnker, Zhi Han, B. Krogh, J. Ouaknine, O. Stursberg, M. Theobald. *Abstraction and Counterexample-guided Refinement of Hybrid Systems.* International Journal of Foundations of Computer Science, Vol 14, Number 3, 2003.

[3] J. Dahmann, R. Fujimoto, and R. Weatherly. *The Department of Defense High Level Architecture* In Proceedings of the 1997 Winter Simulation Conference, pp.142-149, ACM, 1997.

[4] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. *HyTech: A Model Checker for Hybrid Systems.* Software Tools for Technology Transfer 1:110-122, 1997.

[5] B. Selic, G., Gullekson, P. Ward. *Real-Time Object-Oriented Modeling*. John Wiley & Sons, 1994.

[6] *TrueTime* source code and documentation http://www.control.lth.se/~dan/truetime/