

Timing-Focused Design of Embedded Systems

Matthias Dörfel, doerfel@inchron.com
Tapio Kramer, kramer@inchron.com



Timing errors resulting from design faults are often discovered very late during the development process. However, if during the system design the timing properties and requirements (response times and latencies, resource utilization) are taken into consideration, expensive re-designs can be reduced. This presentation describes an approach whereby an UML model is extended to become a timing model which is analyzed at an early stage by means of real-time simulation and analysis.

Motivation

With regard to the real-time behavior of embedded systems in particular, the system architect has to cope with a dilemma: The timing behavior of the software to be implemented later determines considerably how the system architecture is designed, which in turn has implications for the timing behavior. Both affect fundamental questions such as the selection of processors and bus systems and the partitioning of the software modules and their distribution among tasks and processors.

For the functional architecture, it has been realized for a long time that a model-based development process is the method of choice for mastering complexity. If, in addition to this, a modeling technique is chosen which produces executable models even during the early development phases, the function model can be simulated, verified and optimized.

The same advantages are offered by a real-time simulation of the real-time behavior of the embedded system. In addition to the function model, a timing model is created by extending the existing function model with timing and performance information [LOK]. Utilizing this timing model, the architect can then argue his decisions by means of simulations and analyses of the real-time behavior of the planned system architecture.

Modeling in UML

In the model-driven development process (Model Driven Design, MDD), the system is developed using visual representations and the deployment of industrial standards such as SysML or UML. Complex designs can be entered by the appropriate domain experts in the presentation form familiar to them, which ensures that the required functionality will be implemented. The models can be run during the design process. It is also possible to generate a code which corresponds to the specification. Both ensure at an early point in time that the required functions are provided and the required system reactions will occur.

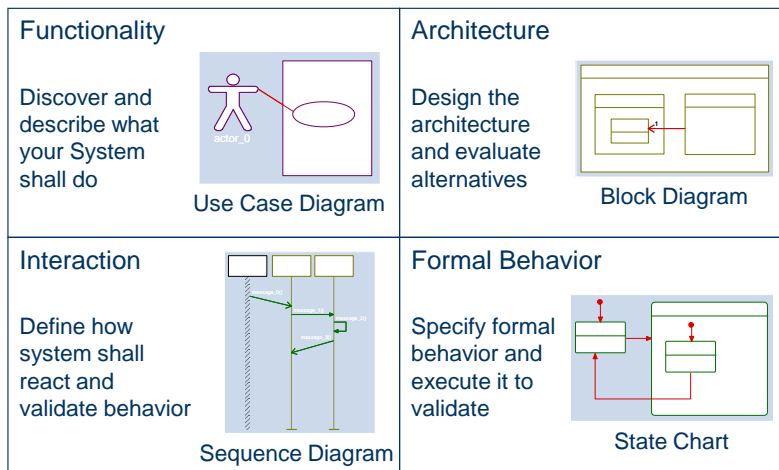


Figure 1: The modeling quadrathlon

The system is graphically modeled in four types of diagrams. The function model (e.g. in a use case diagram) describes what is to be provided by the system. In the structure and block diagrams the system architecture is expressed. The interaction of the system with the environment and the communication between the system components is formulated by means of a sequence diagram. It also shows the specified reaction of the components which can be compared to the result of the simulation. The detailed behavior is specified in the state charts and the activity diagrams. Often also the implementation can be created from this using a code generator.

With these four diagram types, the system can be fully described with all its facets, which in parts are independent of each other. Depending on the purpose of the system and the application, they are often detailed to different degrees. For the real-time properties of the system, in particular the architecture model, the interaction with the system environment and the behavior specification are decisive. For this, attributes are added to these diagrams in order to obtain the timing model.

Modeling of the timing behavior

The extension of the system model to form a timing model is basically carried out in three areas: the description of the timing requirements of the functions, the distribution of the functions to resources, and the definition of system activation.

By annotating the system parts with their net runtimes, the timing behavior of the functions can be determined later from the functional behavior. These parameters are specified as properties of the system parts.

The functions are distributed to the hardware resources (mapping or deployment) in order to define which resource is used for running the function and is using up time there. For the solution presented here, the mapping will be presented similar to SysML as a relation of classes (tasks) to blocks (CPUs). The stereotype used for the CPU offers the option to reference a processor model from the simulation library in the INCHRON Tool-Suite. Essential parameters of these models are the RTOS working on this processor, its scheduling as well as the details of the hardware and processor core. These parameters can also be overwritten in the architecture diagram depending on the model.

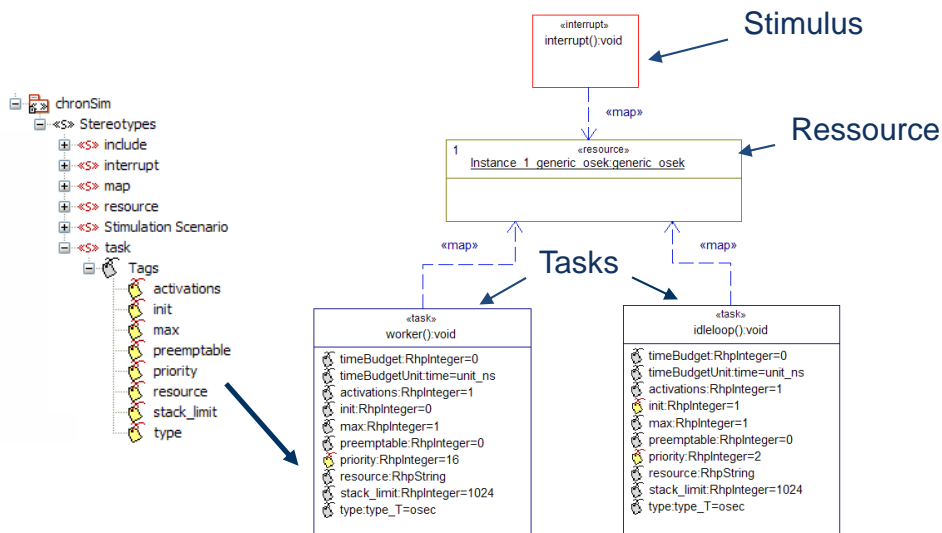


Figure 2: SysML mapping of tasks to processors

In order to model and simulate the dynamic reactions and the dynamic timing behavior of the system, it must be defined how the system is activated from the outside. The activations of embedded systems are usually realized as interrupts and received messages. Interrupts from various sources can have different rates, and they may be correlated with each other. For input, it is useful to continue using scenarios defined in the interaction diagrams. For this, sequence diagrams, for example, are translated into interrupt sequences which can be further varied in the INCHRON Tool-Suite. Furthermore, real-time requirements can be derived from the diagrams which are automatically validated in the simulation.

The result of this addition is a UML real-time simulation model. In addition to the simulation of the function behavior for early functional verification, the real-time behavior and the system performance can be simulated and checked as early as during the architecture phase.

Simulation, analysis and optimization

The described additions of UML have been implemented in a profile in the IBM[®] Rational[®] Rhapsody[®] tool. In contrast to the SPT and MARTE profiles, this contains profiles only for the most necessary stereotypes and types in order to keep the entry threshold low. In addition, the profile also contains the code for code generators.

From the real-time simulation model in UML, Rhapsody automatically generates a project for the real-time simulation and analysis with chronSIM. This project comprises an XML project file and C code (a Task-Model [KOM]) for the detailed modeling of the system model in a form which can be added by the user and which can be as close as desired to the target code.

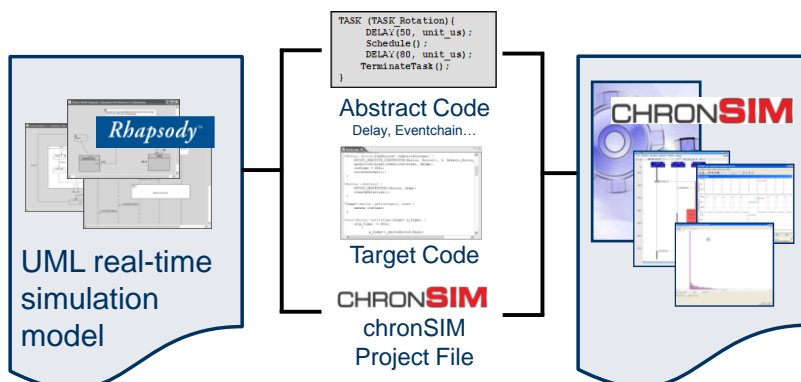


Figure 3: Rhapsody generates a Task-Model for real-time simulation with chronSIM

For the simulation of the real-time behavior the Task-Model and all other information is used to generate a executable simulation model. The user can, during the simulation, interactively enter the defined activations into the simulations. The result is the presentation of all relevant system events and their time of occurrence. This way the events and actions in the individual microprocessors of the system can be evaluated with respect to time.

In the various diagrams of the INCHRON Tool-Suite simulation window the real-time behavior of the system can be observed and analyzed. In the task state diagram it can be observed precisely how the system parts, which have been allocated to tasks and interrupt service routines, are activated depending on the system model and how they preempt each other. If, as shown in Figure 4, a cyclical interrupt activates a task, this sequence is shown in curves 1 and 2 in the diagram. An asynchronous, high-priority task can, however, block the cyclical task for such a long time that the cyclical tasks are activated multiple times. This loss of an activation is recognized by the real-time simulator and is marked in the diagram.

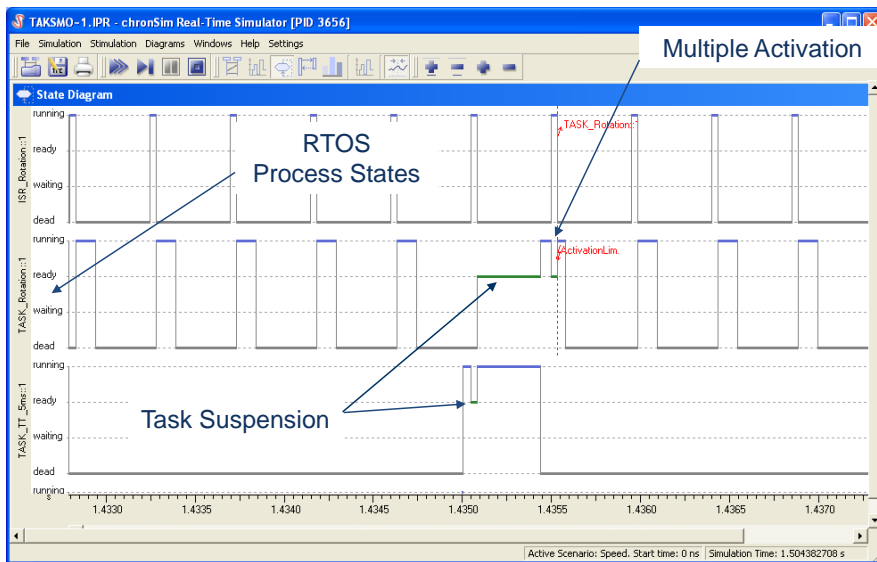


Figure 4: Task state diagram with asynchronous processes

The sequence diagram with an additional time axis (compared to the UML pendant) allows a thorough analysis of the real-time processes. The time axis shows the simulation time for all events. Every task and ISR is displayed with its own lifeline. By a different line width it is symbolized whether the task or ISR is currently running. For every evaluated real-time requirement the status is indicated with red and green boxes.

Summary

For the example of UML, a possibility is shown how an existing system specification can be extended by a model of the temporal behavior of an embedded system. This allows the user at a very early stage in the development process to evaluate the effects of his design decisions on the real-time behavior.

This approach has been implemented as a profile in IBM Rational Rhapsody and is in use by customers. The code generator mechanisms provided in Rhapsody automatically generate a real-time simulation model. The time required for creating a further timing model in addition to the function model is no longer required.

Literature

[KOM] T. Komarek, M. Dörfel, R. Münzenberger; January 2007; Developing Real-Time Constrained Embedded Software Using Task-Models; Advanced Automotiv Electronics (AAE 2007), Gaydon; www.aae-show.co.uk

[LOK] M. Lokietsch; Juni 2009; Performance-Analyse von UML-Systemen; Fachkongress Echtzeitentwicklung; München; www.echtzeitkongress.de

Author:

Matthias Dörfel, founder and director of INCHRON GmbH, studied computer science at the Technical University of Munich and then garnered experiences with the development of embedded systems as a hardware and firmware developer. The return to the university environment as research assistant at the University of Erlangen-Nuremberg resulted in him being introduced to the other founders of INCHRON GmbH during the DFG (German Research Foundation) focus program.



IBM[®], Rational[®] and Rhapsody[®] are registered trademarks of the International Business Machines Corporation. INCHRON[®] and chronSIM[®] are registered trademarks of INCHRON GmbH.