Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 797–804
ISBN 978-83-60810-27-9
ISSN 1896-7094

Integration of Scheduling Analysis into UML Based Development Processes Through Model Transformation

Matthias Hagner and Ursula Goltz
Institute for Programming and Reactive Systems,
TU Braunschweig,
Mühlenpfordtst. 23,
38106 Braunschweig, Germany,
{hagner, goltz}@ips.cs.tu-bs.de

***Abstract*—The complexity of embedded systems and their safety requirements have risen significantly in recent years. Models and the model based development approach help to keep overview and control of the development. Nevertheless, a support for the analysis of non-functional requirements, e.g. the scheduling, based on development models and consequently the integration of these analysis technologies into a development process exists only sporadically. The problem is that the analysis tools use different metamodels than the development tools. Therefore, a remodeling of the system in the format of the analysis tool or a model transformation is necessary to be able to perform an analysis. Here, we introduce a scheduling analysis view as a part of the development model, which is a MARTE annotated UML model to describe a system from the scheduling behavior point of view. In addition, we present a transformation from this annotated UML model to the scheduling analysis tool SymTA/S and a treatment of the analysis results to integrate scheduling analysis into a development process. With our approach it is not necessary to remodel the system in an analysis tool to profit from the analysis and its results. Additionally, we illustrate our approach in a case study on a parallel robot controller.**

I. Introduction

Model based development is widely appreciated in the embedded systems domain to cope with the complexity. As a generic and standardized approach, UML [1] has established as one of the most important notations for modeling software. The MARTE profile (UML Profile for Modelling and Analysis of Real-Time and Embedded systems) [2] makes domain specific ideas, relevant for real-time and embedded systems design, available in a unified modeling framework. Based on this profile, we have defined the UML scheduling analysis view [3] as a part of the design model that concentrates on the scheduling analysis aspects and leaves out all unnecessary information to help the developer focusing on these aspects. This approach is based on the cognitive load theory [4], which states that human cognitive productivity dramatically decreases with the amount of different dimensions to be considered at the same time.

Besides specification and tracing of timing requirements through different design stages, the major goal of enriching models with timing information is to enable early validation and verification of design decisions. As designs for an embedded or safety critical systems may have to be discarded if deadlines are missed or resources are overloaded, early timing analysis has become an issue and is supported by a number of specialized analysis tools like SymTA/S [5], MAST [6], and TIMES [7]. However, the metamodels on which analysis models of the tools are based differ from each other and in particular from UML models used for design, especially the UML scheduling analysis view. Thus, to make an analysis possible and to integrate it into a development process, the developer has to remodel the system in the analysis tool. To avoid this major effort, an automatic model transformation is needed to build an interface that enables automated analysis of a MARTE extended UML model using existing RT analysis technology.

In this paper, we introduce a method to integrate the scheduling analysis into a UML based development process. With our approach it is possible to add scheduling parameters to a UML model and to perform scheduling analysis based on this model. Therefore, we demonstrate a model transformation (realized with the Atlas Transformation Language - ATL [8]) from our proposed UML scheduling analysis view as an extension of Papyrus for UML[1] to the scheduling analysis tool SymTA/S. After the analysis, the results are published in the UML development model to give the developer a feedback concerning the scheduling behavior such that he/she can draw the right conclusions. Consequently, the developer does not need to see SymTA/S or to know how to use the analysis tool to profit from scheduling analysis.

This paper is structured as follows: Section II gives an introduction over the integration of scheduling analysis into the development process, Section III presents the UML scheduling analysis view, in Section IV, SymTA/S

[1]http://www.papyrusuml.org

is explained, Section V and VI present the transformation from the UML scheduling analysis view to SymTA/S and the recirculation and publication of results of the analysis. In Section VII we apply our approach to a case study of our Collaborative Research Centre 562. Section VIII concludes the paper.

II. Integration of Scheduling Analysis into a Development Process

The idea of performing scheduling analysis based on UML models assumes that all the information that is needed for the analysis is already part of the UML model. There is no scheduling analysis tool that is based on UML models or that uses UML models as an input. Therefore, the transformation described in Section V is necessary. But for this transformation, it is necessary that all information needed is already part of the UML development model. If this is the case, the data/necessary information of the system is transformed into the format of the analysis tool.

After the transformation is done, the analysis examines the response times of the tasks and the utilization of the resources. It checks if task chains are executed fast enough or deadlines are missed. Following the analysis, the results are published again in the UML scheduling analysis view (see Figure 1 for the workflow). These steps are done automatically and the developer only has to start the workflow and gets back the analysis results immediately. Afterwards, he/she just has to interpret the analysis results. Consequently, the developer does not need to see or use the analysis tool, as he/she can model and parameterize the system in the UML development model. Even the analysis results are published using UML and the MARTE profile.

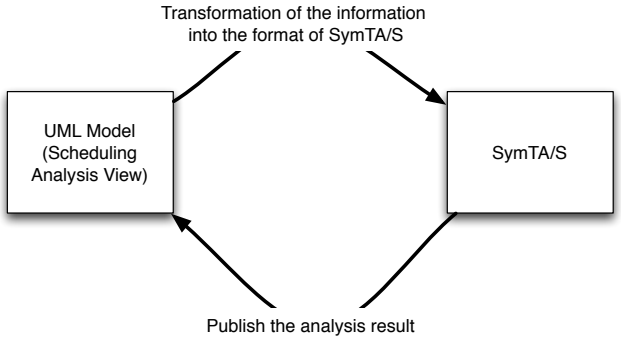

Fig. 1. Illustration of the transformation flow

This approach is independent from the point in time during the development process, as long as the system under consideration is completely described to meet the criteria of a scheduling analysis. It is useful at the beginning of a development process to help making decisions concerning the number of resources or the distribution, even if at this time mainly estimated values must be used and the system is not completely designed (e.g. tasks are still combined and not yet broken down). At a later stage of development, more exact/measured (execution) times can be used to determine more accurate analysis results. Regardless of the development stage, the workflow, described in Figure 1, is the same.

III. The UML Scheduling Analysis View

The UML scheduling analysis view [3] was designed regarding the information required by a number of scheduling analysis tools (e.g. SymTA/S). Therefore, it concentrates on and highlights timing and scheduling aspects. Unlike usual design models, it contains all necessary information for an analysis (priorities, scheduling algorithms, task execution times, etc.).

The view consists of three different UML diagram types and a selection of MARTE stereotypes and tagged values (see Figure 2). The diagram types are class diagrams as an architectural view (to describe the structure, associations, and allocations of the systems' elements), object diagrams as a view on the runtime system (to instantiate the concrete system that should be analyzed based on the architectural view), and activity diagrams as a workload view (to describe workload situations, flows, and dependencies of tasks).

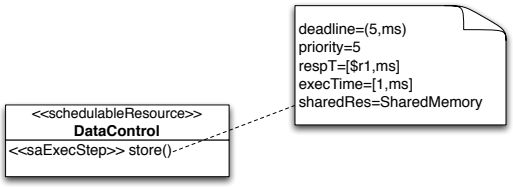

Fig. 2. Example of MARTE stereotypes and tagged values

The stereotypes and the tagged values are based on the MARTE UML profile. We only used a small amount of the defined elements for the scheduling analysis view. One goal of the view is to keep it as simple as possible. Therefore, only elements are used that are necessary to describe all the information that is needed for an analysis. In Table I all used stereotypes and tagged values are presented.

Class diagrams are used to describe the architectural view/the structure of the modeled system. The diagrams show resources, tasks, and associations between these elements. Furthermore, schedulers and other resources, like shared memory, can be defined. Figure 3 shows a class diagram of the scheduling analysis view that describes the architecture of a sample system. The functionalities/the tasks and communication tasks are represented by methods. The tasks are described using the «saExecStep» stereotype. The methods that represent the communication tasks (transmitting of data over a bus) are extended with the «saCommStep» stereotype. The tasks or communication tasks, represented as methods, are part of schedulable resource classes (marked with the «schedulabeResource» stereotype), which combine tasks

TABLE I
ELEMENTS OF THE SCHEDULING ANALYSIS VIEW

Stereotype	used on	Tagged Values
«saExecHost»	Classes, Objects	Utilization, mainScheduler, isSched
«saCommHost»	Classes, Objects	Utilization, mainScheduler, isSched
«scheduler»	Classes, Objects	schedPolicy, otherSchedPolicy
«schedulableResource»	Classes, Objects	
«saSharedResources»	Classes, Objects	
«saExecStep»	Methods	deadline, priority, execTime, usedResource, respT
«saCommStep»	Methods	deadline, priority, execTime, msgSize, respT
«saEndToEndFlow»	Activities	end2endT, end2endD, isSched
«gaWorkloadEvent»	Initial-Node	pattern
«allocated»	Associations	

or communications that belong together, e.g. since they are part of the same use case or all of them are service routines. Processor resources are represented as classes with the «saExecHost» stereotype and bus resources are classes with the «saCommHost» stereotype. The tasks and communications are mapped on processors or busses by using associations between the schedulable resource and the corresponding bus or processor resource. The associations are extended with the «allocated» stereotype. Scheduling relevant parameters (like deadlines, execution times, priorities, etc.) are added to the model using tagged values.

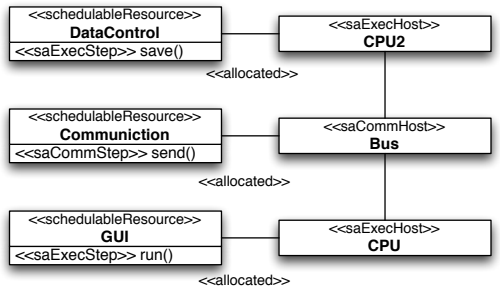

Fig. 3. The architectural view of the scheduling analysis view

The object diagram or runtime view is based on the class diagram/architectural view of the scheduling analysis view. It defines how many instances are parts of the runtime system respectively what parts are considered for the scheduling analysis. It is possible that only some elements defined in the class diagram are instantiated. Furthermore, some elements can be instantiated twice or more (e.g. if a processor is redundant). Only instantiated objects are taken into account for the scheduling analysis and, consequently, for the model transformation.

Activity diagrams are used to describe the behavior of the system. Therefore, workload situations are defined that outline the flow of tasks that are executed during a certain mode of the system. The dependencies of tasks and the execution order are illustrated. The «gaWorkloadEvent» and the «saEnd2EndFlow» stereotypes and their corresponding tagged values are used to describe the workload behavior parameters like the arrival pattern of the event that triggers the flow or the deadline of the outlined task chain. For example, in Figure 4, it is well defined that at first *cpu.run()* has to be completely executed, before *communication.send()* is scheduled etc.. There are restrictions like that no decision nodes are allowed, because the analysis tool SymTA/S does not support these model elements.

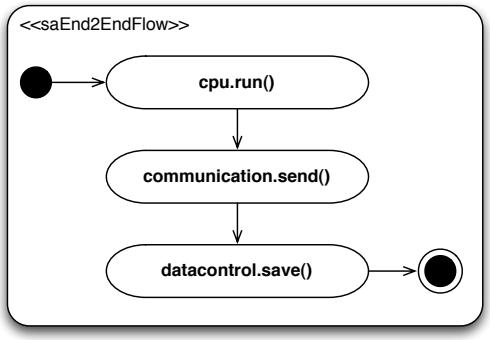

Fig. 4. A workflow description of the scheduling analysis view

The scheduling analysis view can be easily be extended, if necessary. If a scheduling analysis tool offers more possibilities to describe or to analyze a system (e.g. a different scheduling algorithm) and needs more system parameters for it, these parameters have to be part of the scheduling analysis view. Therefore, the view can be extended with new tagged values that offer the possibility to add the necessary parameters to the system description (added to Table I).

Another advantage of using the scheduling analysis view is that the tagged values help the developer to keep track of timing requirements during the development, as these parameters are part of the development model. This especially helps to keep considering them during refinement.

IV. Scheduling Analysis with SymTA/S

We use SymTA/S (**Sym**bolic **T**iming **A**nalysis for **S**ystems) [4] for the scheduling analysis. The example depicted in Figure 5 is the SymTA/S representation of the

system described in Section III and illustrated in Figure 3 and Figure 4. There is one source (trigger), two CPUs (CPU and CPU2), which run two tasks (run and save), and a bus (Bus) with one communication task (send). All tasks are connected using event streams, representing task chains.

SymTA/S links established analysis algorithms with event streams and realizes a global analysis of distributed systems. At first, the analysis considers each resource on its own and identifies the response time of the mapped tasks. From these response times and the given input event model it calculates the output event model and propagates it by the event stream. If there are cyclic dependencies, the system is analyzed from a starting point iteratively until reaching convergence.

V. Transformation and Analysis

During the transformation, the information is taken from the UML metamodel of the scheduling analysis view and is placed in the metamodel of SymTA/S. The scheduling analysis view is modeled in Papyrus for UML and saved as an XMI[9] file. SymTA/S files are also based on XML. Therefore, the ATL transformation [8] has to read the XMI file and create a, for SymTA/S valid, XML file containing the system represented in the scheduling analysis view.

First, the transformation checks whether the model is suitable at all for a transformation. After that, the hardware components are transformed. The transformation searches for all objects with the «saExecHost» stereotype and generates CPU objects for them in the SymTA/S model. The same happens with objects with the «saCommHost» stereotype for busses.

Thereafter, the activity diagrams/workflow descriptions are considered, and, depending on the parameters of the initial nodes, corresponding source elements in the SymTA/S model are created. In the next step, all tasks and communication tasks (all methods with a «saExecStep» or «saCommStep» stereotype) are created as often, as the class/schedulable resource they belong to is instantiated in the scheduling analysis view (e.g. if a class "Calculate" with the «schedulableResource» stereotype has a method *calculateValues()* that has the «saExecStep» stereotype, and the "Calculate" class is instantiated two times in the scheduling analysis view, there will be two calculateValues tasks in the SymTA/S model).

Analyzing the «allocated» stereotypes does the mapping: For example, if there is an «allocated» marked association between a resource and a class with a «schedulabeResource» stereotype, all methods/tasks of the «schedulabeResource» class are mapped on this resource in the SymTA/S model. The corresponding parameters of the tasks (annotated with tagged values in the UML scheduling analysis view) are entered into the SymTA/S model.

In the last step of the transformation, the event streams in SymTA/S are created based on the dependencies described in the activity diagrams/workflow descriptions. The activity diagrams of the UML scheduling analysis view are observed and the dependencies are added to the SymTA/S model. It is only necessary to detect which actions are connected in the UML view and to reproduce these connections using Event Streams in SymTA/S. As SymTA/S does not offer decision nodes, these elements are not supported during the transformation.

After the complete transformation, SymTA/S is used for the analysis. For this, the tool is started with the created XML file as a parameter. SymTA/S analyses the system and writes back the results into the XML file.

VI. Publishing the Analysis Results in the Scheduling Analysis View

After the analysis is finished, the results are published in the scheduling analysis view. The developer gets the information whether there are tasks or paths/task chains that are not schedulable or that miss their deadlines. Some of the tagged values of the scheduling analysis view are used to give the developer a feedback about the analysis results. For example, in Figure 2 the respT tagged value is empty before the analysis and has a variable ($r1), which means that the response time of the corresponding task is entered at this point after the analysis. There are also other parameters, which give a feedback to the developer (see also Table I):

a) respT: The respT tagged values gives a feedback about the response time of the (communication) tasks and is offered by the «saExecStep» and the «saCommHost» stereotype.

b) end2endT: As the respT, the end2endT tagged values offers the response time, in this case of a path/task chain and is offered by the «saEnd2EndFlow» stereotype. It is not a summation of all response times of the tasks that are part of the path, but a worst case calculated response time of the whole path examined by the scheduling analysis tool (for more details see [5]).

c) Utilization: The «saExecHost» and the «saCommHost» stereotype offer a utilization tagged value that gives a feedback about the load of the CPU or the bus. The value is given in percent. If the value is higher than 100%, it is obvious that this resource is not schedulable (and the isShed tagged value is false too), but even if the value is only slightly under 100% , this is a warning for the developer that this resource is nearly overloaded.

d) isShed: This tagged value gives a response whether the tasks mapped on this resource are schedulable or not (false or true) and is offered by the «saExecHost» and the «saCommHost» stereotype. The tagged values are connected to the Utilization tagged value (e.g. if the utilization is higher than 100%, the isShed tagged value is false). This tagged value is also offered by the «saEnd2EndFlow» stereotype. As the «saEnd2EndFlow» stereotype defines parameters for a path/task chain, the

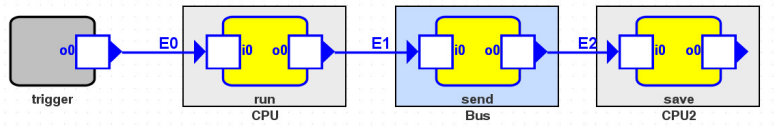

Fig. 5. A sample of a system described in SymTA/S

isShed tagged value gives a feedback whether the deadline for the path is missed or not.

After an analysis is finished, the developer can check if the system is schedulable. For this, the developer checks if the paths/tasks chains are schedulable (isShed tagged value of the «seEnd2EndFlow» stereotype). If this is false, the developer has to find the reason why the scheduling failed. The end2EndT tagged value shows to what extent the deadline is missed, as it gives the response time of the path/task chain. The response times of the tasks and the utilization of the resource give also a feedback where the bottleneck might be (e.g. a resource with a high utilization and tasks scheduled on it with long response times is more likely a bottleneck as resources with low utilizations).

VII. Case Study

The aim of the Collaborative Research Centre 562 (CRC 562)[2] is the development of methodological and component-related fundamentals for the construction of robotic systems based on closed kinematic chains (parallel kinematic chains - PKMs), to improve the promising potential of these robots, particularly with regard to high operating speeds, accelerations, and accuracy [10]. This kind of robots features closed kinematic chains and has a high stiffness and accuracy. Due to low moved masses, PKMs have a high weight-to-load-ratio compared to serial robots. The demonstrators which have been developed in the research center 562 move very fast (up to 10 m/s) and achieve high accelerations (up to 100 m/s^2). The high velocities induced several hard real-time constraints on the software architecture that has been designed to control the robots. The latest version is *PROSA-X* (**P**arallel **R**obots **S**oftware **A**rchitecture - e**X**tended) which uses multiple control PCs to distribute its algorithmic load. *PROSA-X* is a generic architecture which is used to control several different robots. A specifically tailored middleware (*MiRPA-X*) and a bus protocol that operates on top of a FireWire bus (*IAP*) realize communication satisfying the hard real-time constraints [11]. The architecture is based on a layered design with multiple real-time layers within QNX[3] to realize e.g. a deterministic execution order for critical tasks [12].

The robots are controlled using cyclic frequencies between 1 and 8 kHz. If these hard deadlines are missed, this could cause damage to the robot and its environment. To avoid such problems, a scheduling analysis on the basis of models ensures the fulfillment of real-time requirements. The case study presented below has been performed in this context and is based on [13].

In Figure 6 a scheduling analysis view representation of the software architecture, PROSA-X [14], is presented. It consists of a control PC ("Control_PC1"), which performs various computing tasks ("CP1_Tasks"). The control PC is connected via a FireWire (IEEE 1394) data bus with a number of other processors ("DSP_1-7"). The DSPs supervise and control the machine.

The methods represent the following tasks:

- *IAP_D*: This instance of the *IAP* bus protocol receives the *DDTs (Device Data Telegram)* that contain the instantaneous values of the DSP nodes over the FireWire bus.
- *HWM*: The *Hardware Monitoring* takes the instantaneous values received by the *IAP_D* and prepares them for the control.
- *DC*: The *Drive Controller* operates the actuators of the parallel kinematic machine.
- *SMC*: The *Smart Material Controller* operates the active vibration suppression of the machine.
- *IAP_M*: This instance of the bus protocol *IAP* sends the setpoint values, calculated by DC and SMC, to the DSP node.
- *CC*: The *Central Control* activates the currently required sensor and motion modules (see below) and collects their results.
- *CON*: *Contact Planner*. Combination of power and speed control. For the end effector of the robot to make contact with a surface.
- *FOR*: *Force Control*, sets the force for the end effector of the robot.
- *CFF*: Another *Contact Planner*, similar to CON.
- *VEL*: *Velocity Control*, sets the speed for the end effector of the robot.
- *POS*: The *Position Controller* sets the position of the end effector.
- *SAP*: The *Singularity Avoidance Planner* plans paths through the work area to avoid singularities.
- *SEN*: An exemplary *Sensor Module*.

Next we explain the real-time demands on the system. There is no deadline on a specific task, but there are deadlines on task chains/workflows. The first task chain receives the instantaneous values and calculates the new setpoint values (IAP_D, HWM, DC, SMC). The deadline

[2]http://www.tu-braunschweig.de/sfb562
[3]QNX Neutrino is a micro kernel real-time operating system.

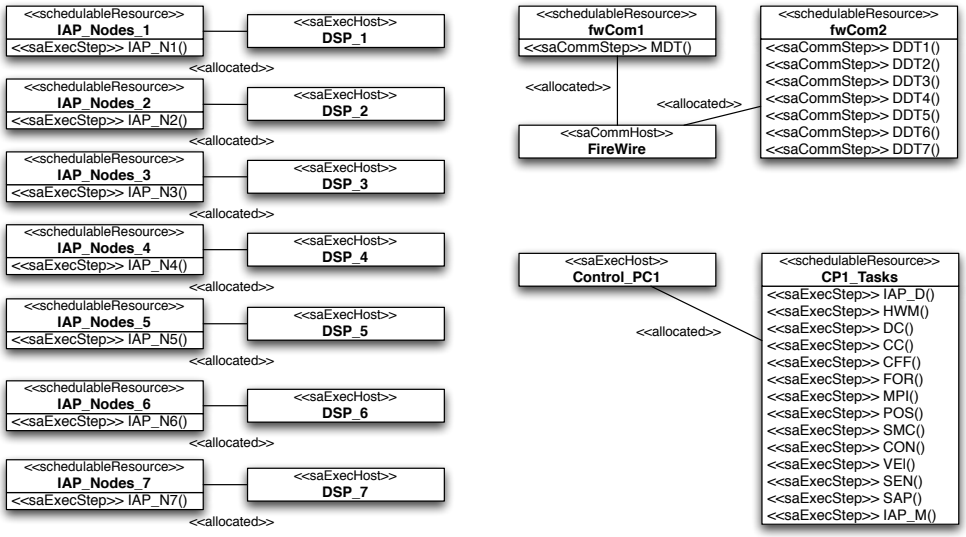

Fig. 6. The architectural view of the PROSA-X system

for this is after 250 microseconds (see Figure 7 for the scheduling analysis view description).

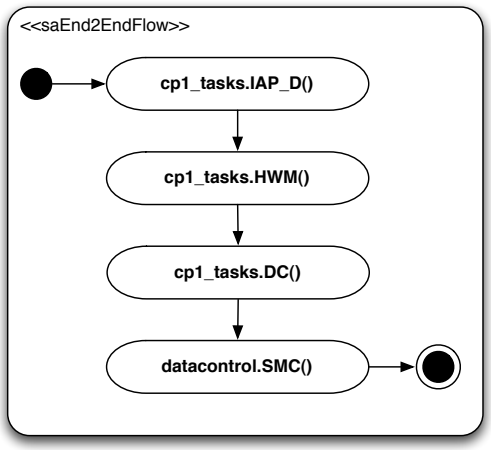

Fig. 7. The receiving of the instantaneous values and the calculation of the new setpoint values

The second task chain contains the sending of the setpoint values to the DSPs and their processing (IAP_M, MDT, IAP_N1, ..., IAP_N7, DDT1, ..., DDT7). This must be finished within 750 microseconds (see Figure 8).

Finally, the third chain comprises the control of the sensor and motion modules (CC, CON, FOR, CFF, POS, VEL, SEN, SAP) and has to be completed within 1945 microseconds (see Figure 9).

The system was modeled in Papyrus for UML as a Scheduling Analysis View. Every element of the architectural view (depicted in Figure 6) is instantiated once in the runtime view.

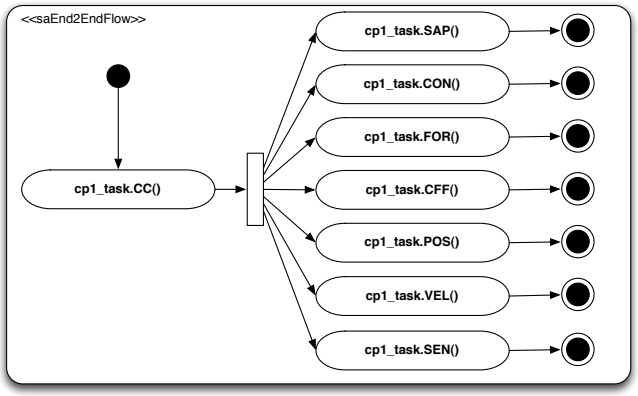

Fig. 9. Control of the sensor and motion modules

The ATL transformation creates a corresponding SymTA/S model and makes it possible to analyze the system. The transformation was successful, the output model was analyzed by SymTA/S and confirms to the expectations: The analysis was successful, all paths keep their real-time requirements, the resources are not overloaded. The SymTA/S model is depicted in Figure 10.

After the successful analysis, the results are published back into the scheduling analysis view (see section VI).

The analysis has confirmed that the system is schedulable. We also integrated this analysis into our approach for finding the best distribution using graph partitioning [15].

VIII. Conclusion and Further Work

We have presented a model transformation from a MARTE annotated UML model to the scheduling analysis tool SymTA/S to make a scheduling analysis based

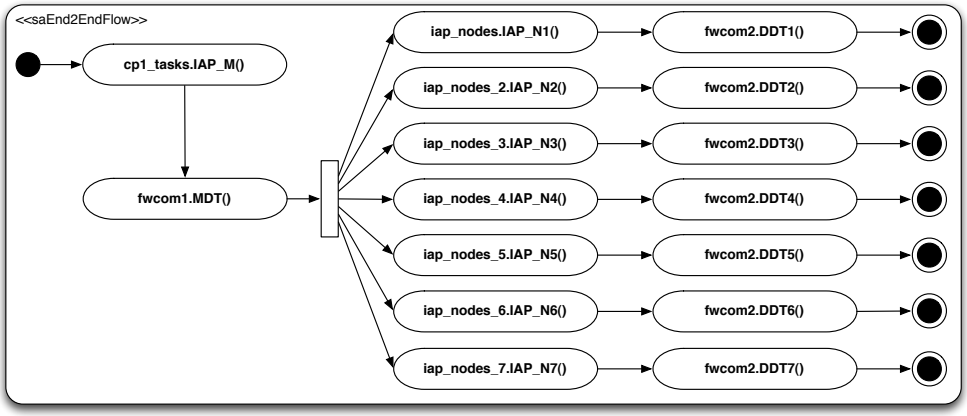

Fig. 8. Sending of the setpoint values to the DSPs

on UML design models possible and to limit the effort for a developer to get feedback concerning the timing/scheduling behavior. We have demonstrated this transformation on a parallel robot controller of our CRC 562. Further steps would be to create more model transformations for other non-functional parameters to corresponding analysis tools.

Another extension of this approach would be to describe a method how to create the scheduling analysis view and how to add the necessary and typically missing parameters to this view to make a scheduling analysis possible (we have made first approaches in [16] and [17]). It is also possible to integrate further tools into the method (e.g. aiT [18]).

Furthermore, a method to help finding the bottleneck if an analysis fails would be of great benefit.

Acknowledgment

The authors would like to thank Symtavision[4] for the grant of free licenses.

References

[1] OMG Object Management Group, "Unified modeling language specification," 2003.
[2] ——, "UML profile for modeling and analysis of real-time and embedded systems (MARTE)," 2009.
[3] M. Hagner and M. Huhn, "Tool support for a scheduling analysis view," in *Design, Automation and Test in Europe (DATE 08)*, 2008.
[4] J. Sweller, "Evolution of human cognitive architecture," in *The Psychology of Learning and Motivation*, vol. 43, 2003, pp. 215–266.
[5] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the SymTA/S approach," *IEEE Proceedings Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, March 2005. [Online]. Available: citeseer.ist.psu.edu/jersak05system.html
[6] M. G. Harbour, J. J. G. García, J. C. P. Gutiérrez, and J. M. D. Moyano, "Mast: Modeling and analysis suite for real time applications," in *ECRTS '01: Proceedings of the 13th Euromicro Conference on Real-Time Systems*. Washington, DC, USA: IEEE Computer Society, 2001, p. 125.
[7] E. Fersman and W. Yi, "A generic approach to schedulability analysis of real-time tasks," *Nordic J. of Computing*, vol. 11, no. 2, pp. 129–147, 2004.
[8] OMG Object Management Group, "MOF 2.0, query / views / transformation ad/2002-04-10, revised submission, version 1.0, 2003/08/18, OpenQVT."
[9] ——, "XML model interchange(XMI)," 1998.
[10] J.-P. Merlet, *Parallel Robots*. Kluwer Academic Publishers, 2000.
[11] N. Kohn, J.-U. Varchmin, J. Steiner, and U. Goltz, "Universal communication architecture for high-dynamic robot systems using QNX," in *Proceedings of International Conference on Control, Automation, Robotics and Vision (ICARCV 8th)*, vol. 1. Kunming, China: IEEE Computer Society, December 2004, pp. 205–210, iSBN: 0-7803-8653-1.
[12] J. Maass, N. Kohn, and J. Hesselbach, "Open modular robot control architecture for assembly using the task frame formalism," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 1–10, 2006, iSSN: 1729-8806.
[13] A. Bragenheim, "Realisierung einer Modelltransformation zur Schedulability-Analyse von UML-Modellen mit SymTA/S," 2009.
[14] J. Steiner, U. Goltz, and J. MaaSS, "Dynamische verteilung von steuerungskomponenten unter erhalt von echtzeiteigenschaften," in *6. Paderborner Workshop Entwurf mechatronischer Systeme*, 2009.
[15] J. Steiner, A. Amado, U. Goltz, M. Hagner, and M. Huhn, "Engineering self-management into a robot control system," in *Proceedings of 3rd International Colloquium of the Collaborative Research Center 562*, 2008.
[16] M. Hagner and M. Huhn, "Modellierung und analyse von zeitanforderungen basierend auf der uml," in *Workshop*, ser. LNI, H. Koschke, Ed., vol. 110, 2007, pp. 531–535.
[17] M. Hagner, M. Huhn, and A. Zechner, "Timing analysis using the MARTE profile in the design of rail automation systems." in *4th European Congress on Embedded Realtime Software (ERTS 08)*, 2008.
[18] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm, "Reliable and precise wcet determination for a real-life processor," in *EMSOFT '01: Proceedings of the First International Workshop on Embedded Software*. London, UK: Springer-Verlag, 2001, pp. 469–485.

[4]http://www.symtavision.com

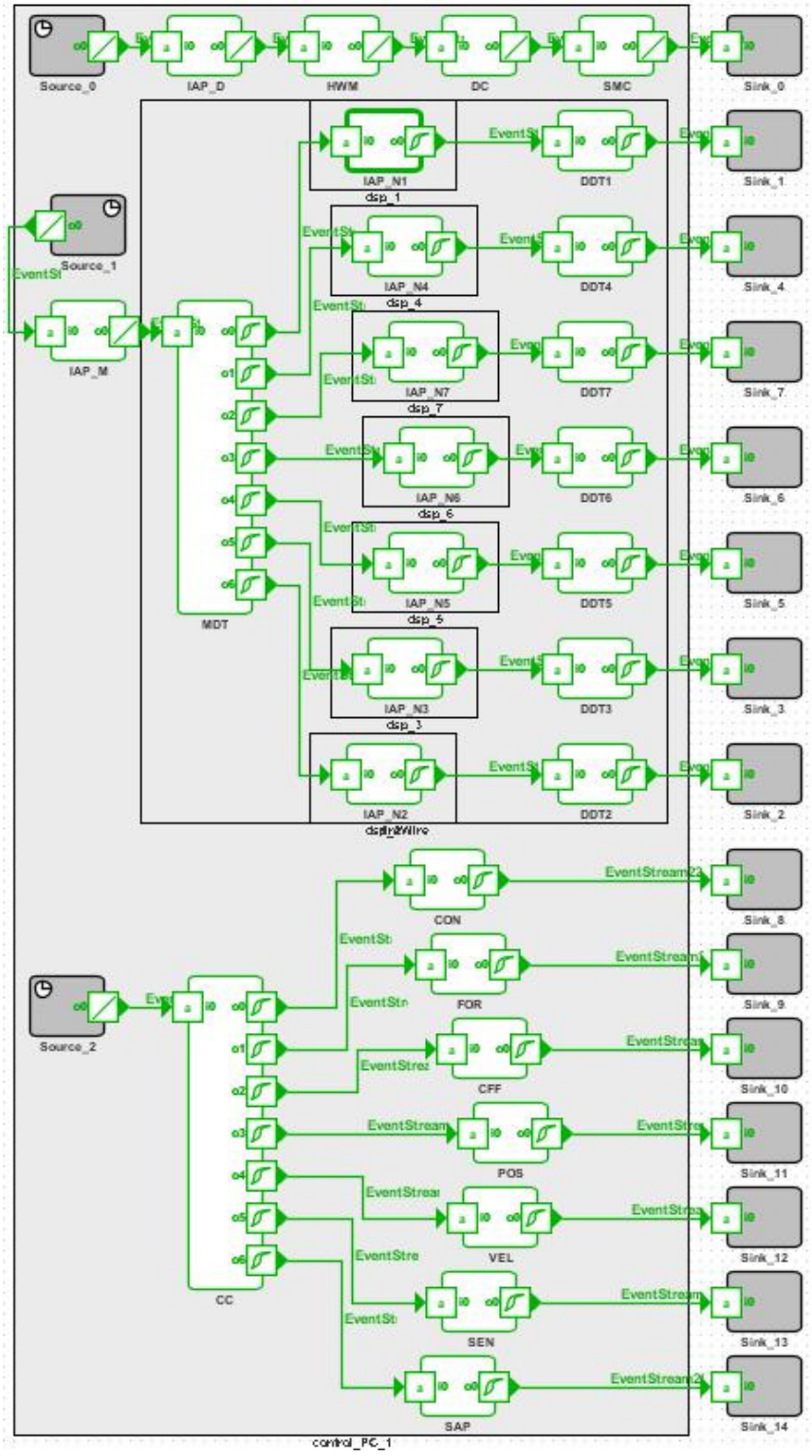

Fig. 10. The SymTA/S description of the PROSA-X system