

An Analysis Tool for UML Models with SPT Annotations

John Håkansson, Leonid Mokrushin, Paul Pettersson, and Wang Yi

Uppsala University
Department of Information Technology
P.O. Box 337, SE-751 05 Uppsala, Sweden
Email: {johnh,leom,paupet,yi}@it.uu.se

Abstract. In this paper, we describe a plug-in for the Rhapsody tool, which demonstrates how UML models with SPT annotations can be analysed using the TIMES tool — a tool for modelling, schedulability analysis, and code generation for timed systems. The plug-in takes as input an UML model consisting of an assembly of components whose behaviours are specified by statecharts. Operations may be annotated with SPT timing parameters for their execution time, deadline, priority etc. The output is a network of timed automata extended with tasks that can be analysed using the TIMES tool. In particular, the TIMES tool will show whether the operations invoked from the UML model are guaranteed to meet their deadlines or not. We describe how this has been done in a case study where an SPT annotated UML model of an adaptive cruise controller is studied.

1 UML with SPT Annotations

We use a subset of UML with SPT annotations (the UML profile for Scheduling, Performance and Time [11]) as an input language for our tool. The models that can be analysed are hierarchical static structure diagrams containing objects representing components or assemblies of components. The behaviour of each component is described by an associated statechart, each executing in its own thread of control. Figure 1 shows an example which can be analysed with the tool presented in this paper. Component A is composed of subcomponents B and C whose corresponding statecharts are shown to the right of the structure diagram.

The operations of components represent tasks being scheduled for execution. We apply the SPT stereotype `SAAction` for operations to annotate them with parameters such as priority, computation time, deadline, etc. Parameters of the operation `foo` of C are shown in Figure 1. Operations are triggered by statecharts, and operations triggered by the same statechart are executed synchronously.

Components can have ports that can be connected with directed links allowing for interaction between them. There are two types of ports, namely data ports and trigger ports. Data ports are defined using `DataPort` interface which has two methods `get` and `set`. Links between data ports are one place buffers that

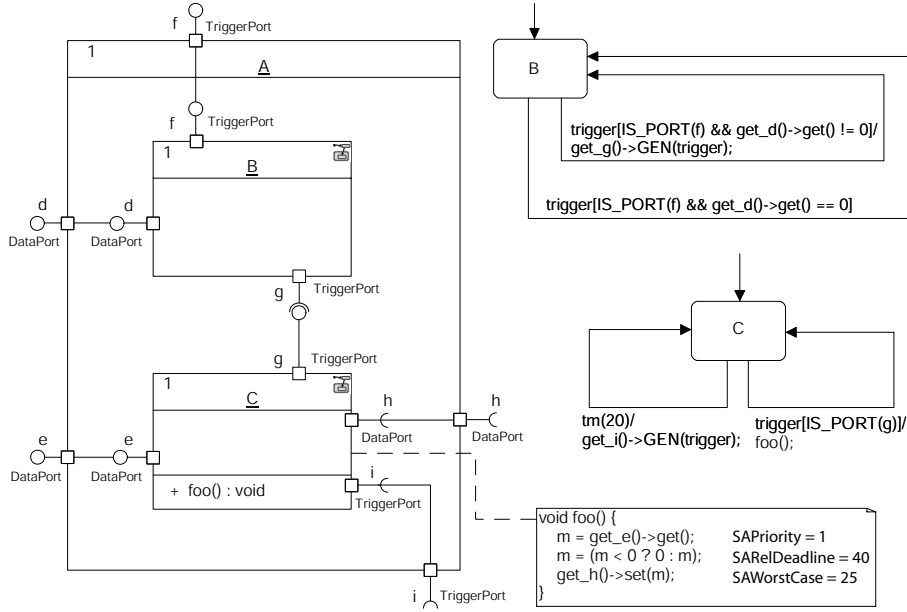


Fig. 1. An example UML model with SPT annotations

can have one sender and many receivers. A data item in a one place buffer is observable by receivers until overwritten by the sender. Trigger ports are defined using `TriggerPort` interface. Untyped events can be sent over the link using the `GEN` method. Events are stored in a buffer of a predefined size until consumed by the receiver.

Every component can be associated with a statechart. Transitions of statecharts are enabled by triggers and/or guards; the associated actions will be executed when the transition is fired. A trigger can be a timeout or the reception of an event from a trigger port. For example in the statechart of component C the left transition has timeout trigger `tm(20)`, and the right transition has event `trigger[IS_PORT(g)]`. The timeout expires 20 ms after state C is reached. Guards are logical conjuncts of integer expression comparisons. Actions can update variables, synchronously execute corresponding operations, generate events on trigger ports, and set the values on data ports.

2 Extracting Timed Models for Analysis

We have developed a tool to extract timed models from UML models with SPT annotations. The extracted model is a network of timed automata [1] extended with tasks [6] and integer variables. Figure 2 shows the timed model extracted from the UML model in Figure 1. The statecharts are converted into timed automata, while the links of the structural diagrams are translated into shared

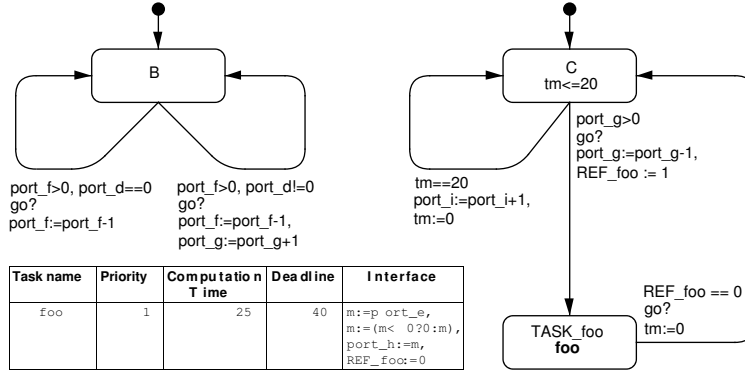


Fig. 2. Timed model extracted from example in Figure 1

data for asynchronous communication between automata. Operations with the stereotype *SAAction* become tasks whose computation time, deadline and priority are given according to the SPT annotations *SAWorstCase*, *SARelDeadline*, and *SAPriority*, respectively.

The ports are represented as shared integer variables that are aliased according to how the links connect them. A variable connecting data ports represent the data item currently residing in the one place buffer. For trigger ports the variable is used to count the number of events queued at the port. The *get* operation on a data port corresponds to accessing the current value of the variable associated with that port, while the *set* operation corresponds to updating the variable with a new value. The *GEN* operation on a trigger port corresponds to increasing the event counter variable, the counter is then decreased when a transition is fired due to an event received at this port.

A transition in a statechart is translated into edges in an automaton, with intermediate locations when needed. An event trigger becomes a guard that is true for non-zero queue lengths of the triggered port. A timeout $tm(T)$ in a statechart i is converted to a guard $tm_i = T$, an invariant $tm_i \leq T$ on the source location, and a reset operation on the clock tm_i for each transition reaching the source location. A call to an *SAAction* operation becomes an intermediate location where the corresponding task is released for asynchronous execution, the automaton will remain in this location until the task completes.

3 Tool Overview

The tool is a plug-in for analysis of Rhapsody [9] UML models with SPT annotations [11] using the TIMES [2] tool. The plug-in can be added to the Rhapsody Tool menu, and uses the Rhapsody COM API to access the UML model. The extracted model is generated as Java objects directly into a running instance of the TIMES tool.

TIMES is a design and analysis tool for real-time embedded systems, based on timed automata [1] extended with tasks [6]. Tasks are temporal abstractions of executable programs represented as their execution time, deadline, priority, etc. Releases of task instances are triggered by timed automata for asynchronous execution according to a given scheduling strategy. The TIMES tool is developed for automated schedulability analysis of the extended models.

4 Case Study

The tools described in this paper have been applied to analyse an adaptive cruise controller (ACC) model. An ACC is an extension of an ordinary cruise controller, with the purpose to control the velocity of a vehicle, while maintaining a minimal distance to any vehicle in front. A radar is used, with an object recognition component, to detect the speed and distance of any vehicle in front.

The ACC model is defined within SaveCCM [8], a component model developed as a part of the SAVE project. The purpose of the project is to develop a component technology for safety-critical vehicle systems. The concepts of data ports and trigger ports in our modelling language comes from SaveCCM. In SaveCCM there are three types of entities, separated using stereotypes: **SaveCOMP** is used to denote components, **Assembly** denotes component assemblies, and **Switch** denotes a special construct for switching triggering events. Statecharts model the triggering of components by first receiving from all input trigger ports, then perform its calculations, and finally generate events on all output trigger ports. Switches differ in that they can generate events on selected output trigger ports.

The ACC assembly of our model is shown in Figure 3. The calculations of each component is modelled by an operation `execute`, with SPT parameters according to Table 1. The ACC is assembled from three components and a sub-assembly. The component `ModeLogics` calculates the current operating mode of the ACC. `ObjectRecognition` applies an algorithm to radar input, in order to determine the velocity and distance of any vehicle in front. `HMI_outputs` determines the output to be sent to the driver panel. `AccControllers` is an assembly consisting of two PID controller components and a switch. The switch determines what controllers are active depending on the current operating mode of the ACC. In standard cruise controller mode only the `SpeedController` is active, but in ACC mode both the `SpeedController` and `DistanceController` are active and coupled in a cascaded control loop.

When the model is exported from Rhapsody into TIMES, the resulting TIMES model is schedulable. The response times reported by TIMES are presented in Table 1. The schedulability analysis was performed in less than one second and consumes 3.3 Mb of memory, using an Intel® Celeron® 1.7GHz computer.

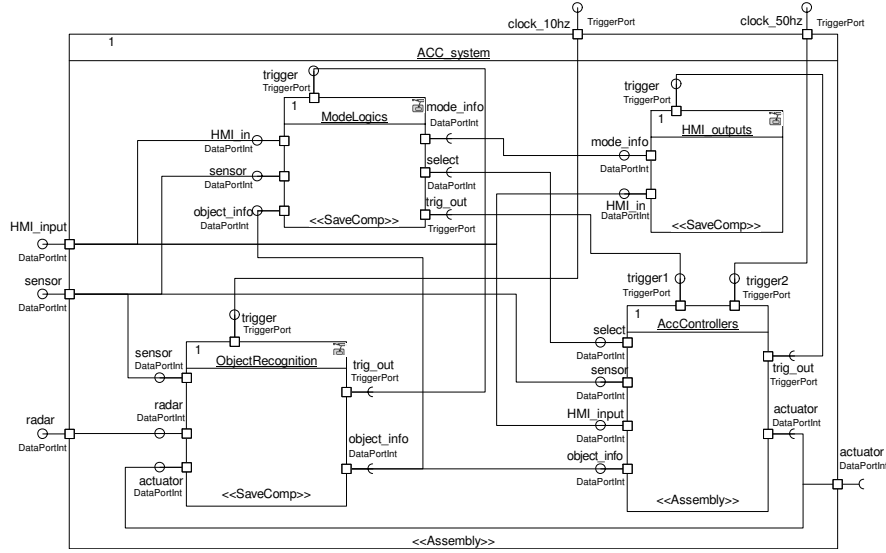


Fig. 3. The ACC subsystem assembly

Component Name	SAWorstCase	SARelDeadline	SAPriority	Worst Case Response Time
SpeedController	5	20	5	5
ObjectRecognition	30	100	4	40
ModeLogics	1	100	3	6
HMI_outputs	2	100	2	2
DistanceController	20	100	1	25

Table 1. The properties of the execute operation for the components.

5 Conclusions

In this ongoing work we demonstrate how UML with the SPT profile can be used to model schedulability problems in a way that is analyzable by the TIMES tool. Some work remains to be done. The case study model should be extended with more details, for a more reliable validation of the work. To handle a richer class of UML models and data structures, we plan to study and extend the tool with predicate abstraction [13, 4, 3].

Related work: Extracting timed models from UML has been done in [7, 5]. There is a tool [12] by OFFIS for analysis of untimed Rhapsody models. Tri-Pacific has a tool [10] for schedulability analysis of periodic task sets extracted from Rhapsody models. In our work we propose a more generic model

where tasks are triggered not only periodically but also by external events. The exact task release pattern is defined by means of UML statecharts.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. TIMES: a tool for schedulability analysis and code generation of real-time systems. In *Proc. of 1st International Workshop on Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science. Springer-Verlag, 2003.
3. Thomas Ball, Rupak Majumdar, Todd D. Millstein, and Sriram K. Rajamani. Automatic predicate abstraction of C programs. In *SIGPLAN Conference on Programming Language Design and Implementation*, pages 203–213, 2001.
4. E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *Proc. of the 12th Int. Conf. on Computer Aided Verification*, pages 154–169, 2000.
5. Alexandre David, Oliver Möller, and Wang Yi. Formal verification of uml statecharts with real-time extensions. In Ralf-Detler Kutsche and Herbert Weber, editors, *Proceedings of FASE 2002*, number 2306 in Lecture Notes in Computer Science, pages 218–232. Springer-Verlag, 2002.
6. Elena Fersman, Paul Pettersson, and Wang Yi. Timed automata with asynchronous processes: Schedulability and decidability. In J.-P. Katoen and P. Stevens, editors, *Proc. of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, number 2280 in Lecture Notes in Computer Science, pages 67–82. Springer-Verlag, 2002.
7. Susanne Graf, Ileana Ober, and Iulian Ober. Model checking of UML models via a mapping to communicating extended timed automata. In S. Graf and L. Mounier, editors, *Proc. of the 11th International SPIN Workshop*, number 2989 in Lecture Notes in Computer Science, pages 127 – 145. Springer-Verlag, 2004.
8. Hans Hansson, Mikael Åkerholm, Ivica Crnkovic, and Martin Törngren. SaveCCM - a component model for safety-critical real-time systems. In *Proc. of Euromicro Workshop on Component Models for Dependable Systems*. IEEE Computer Society Press, 2004.
9. David Harel and Eran Gery. Executable object modeling with statecharts. *Computer*, 30(7):31–42, 1997.
10. Paulo Martins. *Integrating Real-Time UML Models with Schedulability Analysis*. Tri-Pacific Software Inc., 2004.
11. Object Management Group Inc. *UML Profile for Schedulability, Performance, and Time Specification*, 2002.
12. OFFIS. *The Rhapsody UML Verification Environment – Installation-Guide and Tutorial*, 2004.
13. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Proc. of the 9th Int. Conf. on Computer Aided Verification*, volume 1254, pages 72–83. Springer-Verlag, 1997.