Model-based Analysis of Distributed Real-time Embedded System Composition[*]

Gabor Madl[†]
Center for Embedded Computer Systems
University of California, Irvine, CA 92697
gabe@uci.edu

Sherif Abdelwahed
Institute for Software Integrated Systems
Vanderbilt University, Nashville, TN 37205
sherif@isis.vanderbilt.edu

ABSTRACT

Key challenges in distributed real-time embedded (DRE) system developments include safe composition of system components and mapping the functional specifications onto the target platform. Model-based verification techniques provide a way for the design-time analysis of DRE systems enabling rapid evaluation of design alternatives with respect to given performance measures before committing to a specific platform. This paper introduces a semantic domain for model-based analysis of a general class of DRE systems capturing their key time-based performance measures. We then utilize this semantic domain to develop a verification strategy for preemptive schedulability using available model checking tools. The proposed framework and verification strategy is demonstrated on a mission-critical avionics DRE system case study.

Categories and Subject Descriptors

I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

design, verification

1. INTRODUCTION

When dealing with multiple Quality of Service (QoS) properties, DRE system designers often face a gap between design and implementation, since properties are specified in declarative way such as worst case execution times, priorities or in even broader range such as adaptible, fault-tolerant etc. On the contrary, implementations typically follow an imperative approach which makes hard to reconstruct system structure and behavior from the source code. To address these challenges, it is useful to analyze system behavior at design-time, thereby enabling developers to select suitable design alternatives before committing to specific platforms. *Model-based verification* techniques [2] provide a way for the design-time analysis of many engineering systems including DRE systems and allow, through abstraction, to overcome accidental complexities associated with third generation programming languages such as memory management and pointers. Furthermore, they provide a way to observe and prove properties specified in a declarative way by analyzing the composition of imperative component implementations.

In this paper, we extend the earlier works [12, 11] into the generalized DRE SEMANTIC DOMAIN – the basis for the next generation OPEN-SOURCE DREAM [7] – using a formal model of computation (MoC) that captures detailed operation settings, such as asynchronous event channels, as well as essential time-based QoS properties of DRE systems. Based on the proposed extensions, we show how the DRE SEMANTIC DOMAIN can be used to verify the *preemptive* scheduling of distributed multiprocess real-time embedded systems using a novel conservative discrete approximation scheme. The proposed method and case study are explained in greater detail in [10].

2. THE DRE SEMANTIC DOMAIN

In this section we introduce a computational model that can express the event-driven nature of DRE systems. The detailed formal model is presented in [10]. We define a model on a distributed platform with possible execution preemptions. DRE system models can be built by the composition of these components.

2.1 The Timer

The timer is a simple periodic event generator which releases task initiation events at a specified rate. Timers may represent sensors sampled at a predefined rate and are modeled by a generic timed automaton model shown in Figure 1.

2.2 The Task

Tasks are the main components which describe actors in DRE systems. In addition to the states specifying the execution of the task (`idle`, `enabled`, `executing`, `preempted`), we also introduce a `timeout` state which can be used to express undesired behavior of the system such as when the task cannot finish the execution before its respective deadline.

[*] This research was supported by the NSF ITR Grant CCR-0225610 "Foundations of Hybrid and Embedded Software Systems."

[†] Work done while at the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37205

EMSOFT'05, September 19–22, 2005, Jersey City, New Jersey, USA.
Copyright 2005 ACM 1-59593-091-4/05/0009 ...$5.00.

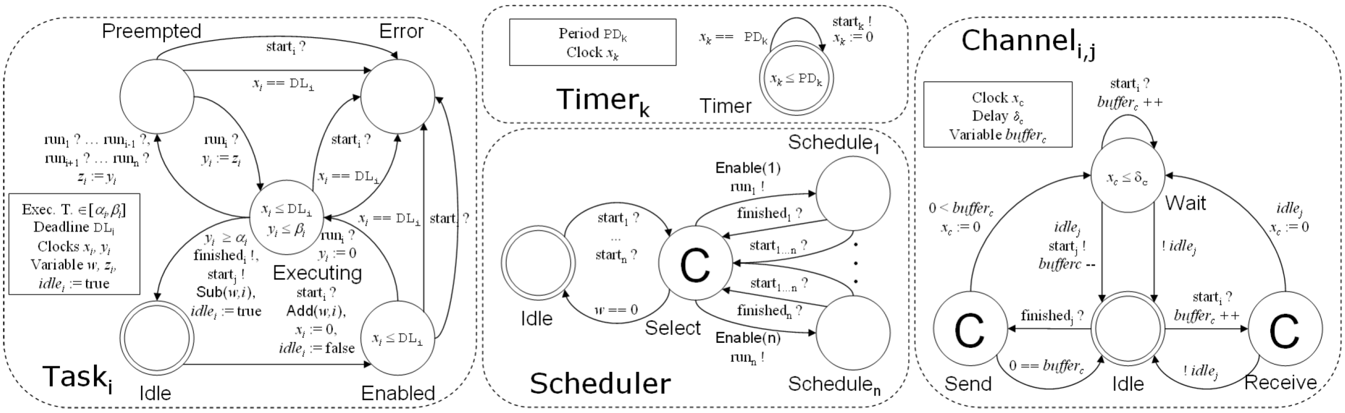

Figure 1: Generic Model of the DRE SEMANTIC DOMAIN

2.3 The Event Channel

Event propagations between tasks follow a non-blocking broadcast semantics – events which are not received are lost. We assume that this is the only case when events get lost. Event channels provide the necessary mechanism to allow reliable asynchronous communication – the publisher does not have to block until the consumer is ready to receive the event. Published events are stored in the event channel until the consumer is ready to receive them. States denoted with the **C** letter are *committed* states. This expresses time constraints, an outgoing transition has to be taken instantaneously, otherwise the system will deadlock. Event channels are generically represented by the timed automaton shown in Figure 1. Note that this model takes into account possible communication delay as represented by the channel dependent maximum delay factor δ_c.

2.4 The Scheduler

To express the mapping of execution tasks to platform processors we introduce the scheduler modeling construct shown in Figure 1. The scheduler selects enabled tasks for execution and triggers preemptions according to the scheduling policy.The scheduler initially starts in the `idle` state. It will move to the `select` state if any tasks become eligible for execution. The selection is made instantaneously from the of enabled tasks' queue and the selected task will receive the `run` event which triggers its execution. The scheduler moves to the `idle` state if no task is ready for execution.

The scheduling policy is encoded in three functions: (1) $\mathsf{Add}(w, i)$, which increases the current priority level when task$_i$ becomes ready, (2) $\mathsf{Sub}(w, i)$, which decreases the current priority level when task$_i$ becomes ready, and (3) $\mathsf{Enable}(w, i)$, which evaluates to `true` if the ith task is eligible for execution. For example, in the case where priority is directly proportional to the component index, $\mathsf{Add}(w, i) = w + 2^{i-1}$, $\mathsf{Sub}(w, i) = w - 2^{i-1}$, and $\mathsf{Enable}(w, i) = 2^{i-1} \le w < 2^i$. Other scheduling schemes can be established by defining appropriate formulas for the three functions outlined above.

2.5 Preemption Model Using Discretized Time

The proposed model of computation corresponds to the stopwatch model due to assignments of variables to clocks. Deciding the preemptive schedulability of the stopwatch model has been shown to be undecidable using timed automata [9]. To address this issue, we implement a discretized preemption scheme in which task interruption can only be granted at specific intervals during the task execution. We implement non-blocking preemptions – after the preemption the task resumes the execution from the last discrete checkpoint. The precision can be set independently for every UPPAAL `Task` model using the `granularity` parameter. Non-blocking preemptions imply longer execution time than the parameter `WCET` time in the preempted tasks. This method gives an overapproximation of the safe states of the system, when the schedulable discretized approximation implies a schedulable system in continuous time.

3. PROBLEM FORMULATION

This paper considers the problem of deciding the schedulability of a given set of tasks with event- *and* time-driven interactions, on a distributed preemptive platform as described in Section 2.

DEFINITION 1. *The system is schedulable if all tasks finish their execution before their respective deadlines.* □

We use a timed automata formulation of the problem which translates the schedulability problem into a reachability problem in which the set of tasks are schedulable if a predefined `error` state is not reachable in any of the tasks' timed automata.

DEFINITION 2. *We define a frame period for a task $t \in T$, referred to as $\mathsf{Period}(t)$ as the period of the slowest timer on which the task depends.* □

Tasks that are assigned to the same platform processor and have the same frame period are in the same frame. Therefore, the set of frames is a partition of the set of tasks T. For a task t we write $\mathsf{Frame}(t)$ to identify the set of all tasks in T belonging to the frame of t.

THEOREM 1. *If the system is schedulable using discretized time preemptions it is also schedulable in continuous time if*

$$(\forall t \in T) \quad \mathsf{D}(t) \ge \mathsf{Period}(t) - \sum_{t' \in \mathsf{Frame}(t) - t} \mathtt{WCET}(t')$$

where $\mathsf{D}(t)$ *denotes the deadline of the task and* $\mathtt{WCET}(t)$ *denotes the worst case execution time of the task.*

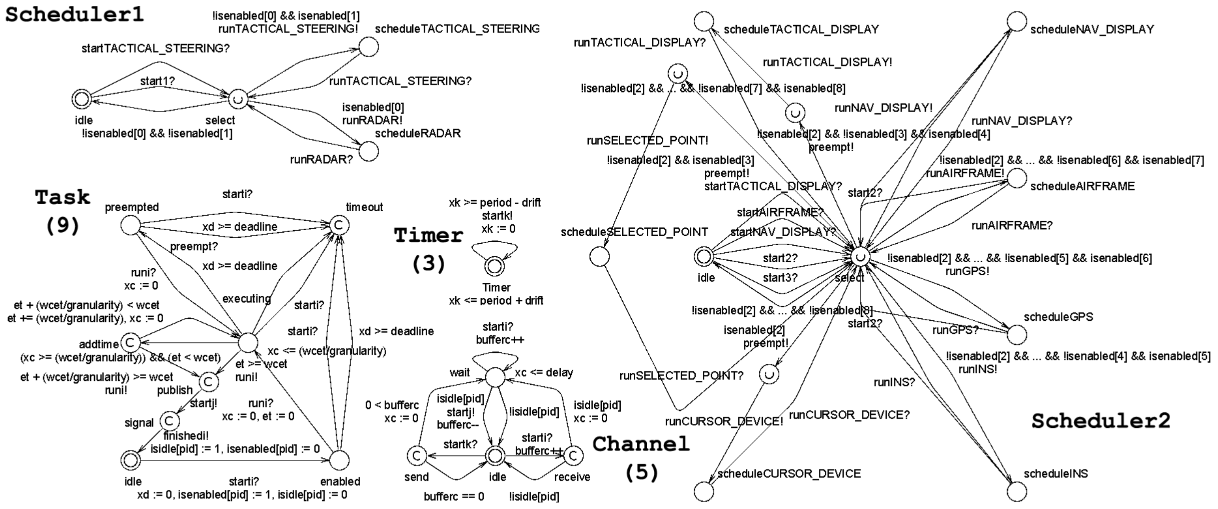

Figure 2: Uppaal Timed Automata Models

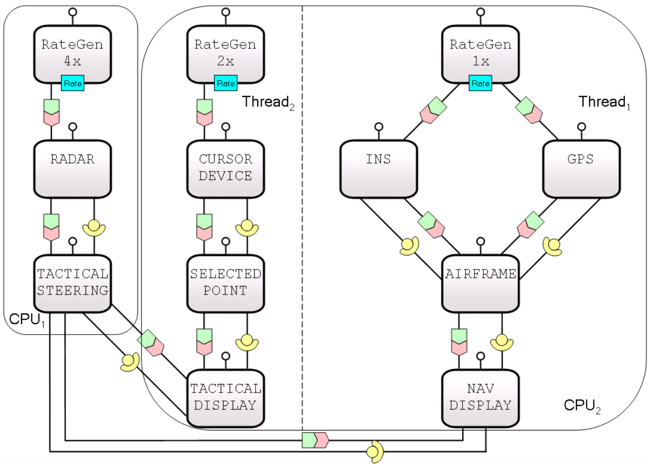

Figure 3: Bold Stroke application deployed on a preemptive multiprocessor platform

The proof, based on the formal specification of the DRE Semantic Domain, is presented in [10].

4. CASE STUDY

In this section we show how we applied model-based verification to a case study of a representative DRE system from the domain of avionics mission computing. Figure 3 shows the component-based architecture of the system, which is built on the Boeing Bold Stroke real-time middleware [14].

The application is deployed on a preemptive multiprocessor platform. The RateGen 4x, RADAR and TACTICAL_STEERING components are deployed on the same application server (CPU$_1$) and are scheduled nonpreemptively. The other application server (CPU$_2$) is a multi-threaded server, where priorities are assigned to the threads. The RateGen 2x, CURSOR_DEVICE, SELECTED_POINT and TACTICAL_DISPLAY components are executed within the context of the higher priority Thread$_2$, the RateGen, INS, GPS, AIRFRAME and NAV_DISPLAY components are executed within the context of the lower priority Thread$_1$. Event propagations are denoted with small arrows in Figure 3 which represent the binding between event sources and sinks. Facets are represented by circles which connect to the receptacles.

4.1 Compositional Analysis Using Uppaal

We use the Uppaal model checker tool [13] for schedulability analysis. Systems in Uppaal are modeled as a slightly modified variant of timed automata and the specification is expressed in a restricted version of the *timed computational tree logic* (TCTL) [1], which is temporal logic that can formalize statements about system models.

Figure 2 shows how we modeled the system in the Uppaal model checker tool. The application consits of 12 components. RateGen components are modeled by Timers, the other 9 components are modeled using Tasks in the DRE Semantic Domain. We can abstract out some of the real-time event channel threads as discussed in [12, 11], to reduce the state space. We have to model event channels explicitly (1) when we have to buffer events or (2) on remote event channels which have measureable delays.

The scheduling policies are represented by Schedulers in the DRE Semantic Domain. Since the Bold Stroke application is deployed on a two-processor architecture we define two schedulers as shown on Figure 2. The design models (such as the model on Figure 3) can be used to generate the scheduling policies as shown in [12, 11].

The analysis models for an application can be built by composing the elements defined in Section 2. The composition is achieved by creating the dependencies using events. Priorities follow dependencies and are therefore fixed. We assign deadlines according to Theorem 1. Table 1 shows the parameter values.

We assign the granularity parameters to 1 millisecond precision in the preempted tasks as shown in Table 1. This analysis has shown that the system is schedulable with the parameters given in 1. We have checked finite buffer sizes with the following TCTL formula: A[] (*Channel*.bufferc < *Channel*.lambdac). Uppaal produces a counter-example for invalid properties, which helps identifying the source

Table 1: Parameter Values for the Application Shown in Figure 3

Task	P	S	WCET	D	G
RADAR	HIGH	HIGH	12	84	1
TACT_ST...	HIGH	MEDIUM	16	88	1
CURS_D...	HIGH	HIGH	18	155	1
SEL_P...	HIGH	MEDIUM	24	161	1
TACT_D...	HIGH	LOW	21	158	1
INS	LOW	HIGH	32	872	32
GPS	LOW	HIGH	29	869	29
AIRFRAME	LOW	MEDIUM	80	920	80
NAV_D...	LOW	LOW	19	859	19

of undesired behavior. Finally, we checked that eventually every task will execute with: E<> *Task*.executing.

5. RELATED WORK

A generic form to analyze scheduling behavior based on the timed automata model was proposed in [4] for single processor scheduling using the Immediate Ceiling Priority protocol and the EDF algorithm. This model does not have a component model and does not support asynchronous event passing.

The Virginia Embedded Systems Toolkit (VEST) [8] is a framework designed for the reliable and configurable composition and analysis of component-based embedded systems. VEST applies key checks and analysis but does not support formal proof of correctness.

The Automatic Integration of Reusable Embedded Systems (AIRES) tool extracts system-level dependency information from the application models. It performs real-time analysis [5] using Rate Monotonic Analysis techniques.

The Cadena [6] framework is an integrated environment for building and analyzing CORBA Component Model (CCM) based systems. The emphasis of verification in Cadena is on software logical properties. The generated transition system does not represent time explicitly and requires the modeling of logical time that does not allow quantitative reasoning.

Time Weaver (Geodesic) [3] is a component-based framework that supports the reusability of components across systems with different para-functional requirements. It supports code generation as well as automated analysis. It performs model-based Rate-Monotonic Analysis by real-time model checker tools such as TimeWiz®.

6. CONCLUDING REMARKS

This paper presents the DRE Semantic Domain used in the next generation Open-Source Dream [7] model-based verification and analysis framework. We have shown how to analyze and verify the *preemptive scheduling* and *composition* of real-time embedded systems on unsynchronized distributed platforms and how model-based verification provides a way to bridge the gap between *declarative specification* and *imperative implementation*. Our results show that this approach captures *delays* and *drifts* which are common in *unsynchronized reactive real-time systems*. The verification is automatic, exhaustive, and capable of producing counter-examples that helps pinpoint sources of undesired behavior. The method is explained in greater detail in [10].

7. REFERENCES

[1] E. Clarke and E. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. *Logic of Programs, Lecture Notes in Computer Science*, 131:52–71, 1981.

[2] E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D. E. Long, K. L. McMillan, and L. A. Ness. Verification of the futurebus+ cache coherence protocol. In *CHDL '93: Proceedings of the 11th IFIP WG10.2 and in cooperation with IEEE COMPSOC*, pages 15–30. North-Holland, 1993.

[3] D. de Niz and R. Rajkumar. Time Weaver: A Software-Through-Models Framework for Real-Time Systems. In *Proceedings of LCTES*, 2003.

[4] T. Gerdsmeier and R. Cardell-Oliver. Analysis of Scheduling Behaviour using Generic Timed Automata. 42, 2001.

[5] Z. Gu, S. Wang, S. Kodase, and K. G. Shin. An End-to-End Tool Chain for Multi-View Modeling and Analysis of Avionics Mission Computing Software. In *Proceedings of Real-Time Systems Symposium*, 2003.

[6] J. Hatcliff, X. Deng, M. B. Dwyer, G. Jung, and V. P. Ranganath. Cadena: An Integrated Development, Analysis, and Verification Environment for Component-based Systems. In *Proceedings of International Conference on Software Engineering*, 2003.

[7] http://dre.sourceforge.net. Distributed Real-time Embedded Analysis Method. 2005.

[8] J.A. Stankovic and R. Zhu and R. Poornalingham and C. Lu and Z. Yu and M. Humphrey and B. Ellis. VEST: An Aspect-based Composition Tool for Real-time Systems. In *Proceedings of the IEEE Real-time Applications Symposium*, 2003.

[9] P. Krčál and W. Yi. Decidable and Undecidable Problems in Schedulability Analysis Using Timed Automata. In K. Jensen and A. Podelski, editors, *Proc. of TACAS'04, Barcelona, Spain.*, volume 2988 of *Lecture Notes in Computer Science*, pages 236–250. Springer–Verlag, 2004.

[10] G. Madl and S. Abdelwahed. Formal Verification of Distributed Preemptive Real-time Scheduling. Technical report, ISIS, Vanderbilt University, http://www.isis.vanderbilt.edu/publications.asp, 2005.

[11] G. Madl, S. Abdelwahed, and G. Karsai. Automatic Verification of Component-Based Real-Time CORBA Applications. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pages 231–240, December 2004.

[12] G. Madl, S. Abdelwahed, and D. C. Schmidt. Verifying Distributed Real-time Properties of Embedded Systems via Graph Transformations and Model Checking (invited paper, submitted). *The International Journal of Time-Critical Computing*, 2005.

[13] P. Pettersson and K. G. Larsen. Uppaal2k. *Bulletin of the European Association for Theoretical Computer Science*, 70:40–44, feb 2000.

[14] D. C. Sharp and W. C. Roll. Model-Based Integration of Reusable Component-Based Avionics Systems. In *Proceedings of the Workshop on Model-Driven Embedded Systems in RTAS 2003*, May 2003.