In proceedings of Embedded Systems and Applications (ESA)
at WORLDCOMP Jul 2008

Achieving Industrial Strength Timing Predictions of Embedded System Behavior

Mikael Nolin, Jukka Mäki-Turja and Kaj Hänninen
Mälardalen Real-Time Research Centre (MRTC), Box 883, Västerås, Sweden

***Abstract*—** *This paper discusses why the extensive scientific results on predicting embedded systems temporal behavior never, or very seldom, reaches the industrial community. We also point out the main issues that the scientific community should focus on in order to facilitate industrial-strength timing predictions.*

The core problem is that the scientific community uses too simplistic or research oriented timing models. The models stemming from academy do not fit well with the structure of real systems. Thus, extracting a timing model that is amenable for analysis may prove prohibitively difficult. And even if a model can be extracted, it may not capture real system scenarios well. Thus, results from analyzing these models do not reflect real system behavior, leading to unnecessary pessimistic timing predictions.

In recent years, response-time analysis has matured to a degree where models can express complex system behaviors and analysis results are relatively tight with respect to real system behavior. However, in order to fully reach its potential, and be accepted by industry, several improvements of the technique are needed. First, behaviors that are commonly used in industrial systems (such as message passing and client/server-patterns) must be adequately captured by the timing models. Second, unnecessary pessimism in the analysis must be removed (i.e. the analysis results must correlate well with actual system behavior by providing minimal overestimation). Third, correlated behaviors of different parts of the systems must be accounted for (i.e. not all tasks will experience the worst case execution times at the same time).

Keywords schedulability, response time, fixed priority systems

1. Introduction

In this paper we discuss the flagrant discrepancy between academic results and industrial needs within the area of schedulability analysis of real-time systems. The importance of schedulability analysis of real-time systems is quickly increasing. Today, almost all electrical products of some complexity are controlled by an embedded computing system. Often, these products need to interact with an environment in a timely manner, i.e. the computer system is a real-time system. Furthermore, a large class of embedded real-time systems are also safety critical, meaning that a system failure can have potentially catastrophic consequences. These safety-critical real-time systems are found, for instance, in vehicles, robotics, medical appliances, and production facilities.

For safety-critical computer systems, the society is increasing the pressure on system providers to provide evidence that the system if safe. This paper will not dwell into the many important issues of demonstrating safety of a computer controlled systems, e.g. as mandated by the safety standard IEC 61508 [8]. However, one important activity in order to establish the safety of a real-time system is to provide evidence that actions will be provided in a timely manner (e.g. each actions will be taken at a time that is appropriate to the environment of the system). For systems consisting of multiple concurrent/semi-concurrent operating-system tasks the number of possible execution scenarios for each actions is daunting [26], and effectively prohibits testing as a means for verifying the correct timing of actions.

To complement testing, and to provide stronger evidence for correct timing, academia has developed techniques to make *a priori* analysis to verify that each action in a system will be performed before its deadline. These, so called, schedulability analysis techniques have been continuously developed over almost four decades [23]. So, from an academic point of view, schedulability analysis can be considered as a mature technology. However, studying the industrial penetration of schedulability gives a very disappointing image. It is very difficult to find reports of successful use of schedulability analysis in real industrial systems. In fact, it is probably easier to find documents of negative results of trying to use schedulability analysis in industrial systems [15], [32].

In this paper we argue that there are two main reasons for the industrial failure of schedulability analysis:

1) Lack of capabilities in the models used by schedulability analysis techniques, and
2) lack of precision in analysis techniques.

One candidate technique that could resolve the above obstacles is schedulability analysis based on timed-automata models [1]. However, for systems of non-trivial size the analysis time, and required memory resources, is overwhelmingly large. Hence, with respect to current limitations in analysis of timed-automat models, this technique does not currently provide a viable path towards industrial-strength schedulability analysis. Thus, we have to find techniques that address the above deficiencies without incurring too large costs in terms of computing resources.

Amongst the more traditional, analytical, schedulability techniques, the response-time analysis of tasks with offsets (RTA), introduced in section 2, stands out as the prime candidate with respect to the two main obstacles above. That is, already today, the models used by RTA have the ability to model quite complex system behaviors, and the precision offered by RTA is the best available amongst analytical techniques. In the rest of this paper we will describe the state-of-the-art with respect to RTA and concretize the main problems that remain to be resolved in order to resolve the above two obstacles, and thus make RTA a viable solution to industrial use of schedulability analysis.

2. RTA History, background and state of the art

Response-Time Analysis (RTA) [3], [23] is a powerful and well established schedulability analysis technique. RTA is a method to calculate upper bounds on response-times for tasks in real-time systems. In essence RTA is used to perform a schedulability test, i.e., checking whether or not tasks in the system will satisfy their deadlines. RTA is applicable for, e.g., systems where tasks are scheduled in priority order which is the predominant scheduling technique used in real-time operating systems today. Furthermore, RTA is not only used as a schedulability analysis tool, but it is also used in a wider context. For example, schedulability analysis is performed in the inner loop of optimization or search techniques such as task attribute assignment and task allocation. In conclusion, RTA provide the basis of many different analysis and optimization techniques for real-time systems.

Liu and Layland [16] provided the theoretical foundation for analysis of fixed priority scheduled systems. Joseph and Pandya presented the first RTA [12] for the simple Liu and Layland task model which assumes independent periodic tasks. Since then RTA has been applied and extended in a numerous ways, e.g., [4], [5], [10], [13], [14], [21], [24], [25], [27], [29], [30], [31]. These works include lifting the independent task assumption, analyzing communication networks, fault tolerant systems, distributed systems, modeling OS overhead etc. So from a scientific perspective RTA has become a well established and mature technology. A more detailed discussion of some of these improvements can be found in "A Practitioners Handbook for Real-Time Analysis" [11]. This book is focused on a practitioner's point of view and thus aims at applying RTA in an engineering context. A historical perspective of real-time scheduling research, where RTA is a big part, can be found in [23].

2.1. Task model with offsets

Incorporating many of the above research results and extended it further, the task model with offset, or transactional task model [18], [19], [28], can be viewed as the state of the art task model for RTA in the sense that it can handle many different and complex system parameters with few constraints as well as applied for different system models. System parameters include arbitrary deadlines, release jitter, temporal dependencies through offsets, access to shared resources. The RTA has been extensively applied to two different contexts, or system models, holistic analysis for distributed systems [28], [19], hybrid static and dynamic scheduling [17]. Furthermore the task model can also be applied for modeling self suspending tasks or generally any system where tasks may have temporal dependencies (offsets) among them. The formal system model used is as follows:

$$\begin{aligned}\Gamma &:= \{\Gamma_1, \ldots, \Gamma_k\}\\ \Gamma_i &:= \langle\{\tau_{i1}, \ldots, \tau_{i|\Gamma_i|}\}, T_i\rangle\\ \tau_{ij} &:= \langle C_{ij}, O_{ij}, D_{ij}, J_{ij}, B_{ij}, P_{ij}\rangle\end{aligned}$$

The system, Γ, consists of a set of k transactions $\Gamma_1, \ldots, \Gamma_k$. Each transaction Γ_i is activated by a periodic sequence of events with period T_i (or the minimum inter-arrival time between two consecutive events). The activating events are mutually independent, i.e., phasing between them is arbitrary. A transaction, Γ_i, contains $|\Gamma_i|$ tasks, and each task may not be activated (released for execution) until a time, *offset*, elapses after the arrival of the external event.

We use τ_{ij} to denote a task. A task, τ_{ij}, is defined by a worst case execution time (C_{ij}), an offset (O_{ij}), a deadline (D_{ij}), maximum release jitter (J_{ij}), maximum blocking from lower priority tasks (B_{ij}), and a priority (P_{ij}). There are no restrictions placed on offset, deadline or jitter, i.e., they can each be either smaller or greater than the period.

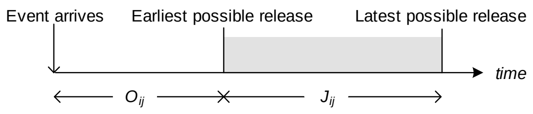

Fig. 1. Relation between an event arrival, offset, jitter and task release

The relation between event arrival, offset, jitter and task release is graphically visualized in Fig. 1. After the event arrival, task τ_{ij} is not released for execution until its offset (O_{ij}) has elapsed. The task release may be further delayed by release jitter (maximally until $O_{ij} + J_{ij}$) making its exact release uncertain. Parameters for an example transaction (Γ_i) with two tasks (τ_{i1}, τ_{i2}) are depicted in Fig. 2.

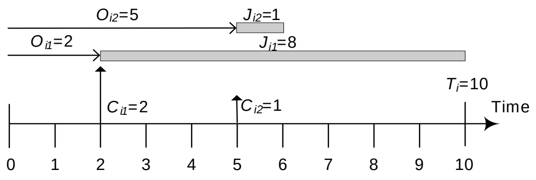

Fig. 2. An example transaction Γ_i

Distributed systems with precedences: The original, and most widely adopted, application for task model with offset is to model precedence relations among tasks in distributed systems [28], [19]. A transaction represents a group of tasks, allocated to several nodes, where every task has a precedence relation to previous task in the transaction. RTA is applied to each node and also the network to produce a holistic schedulability analysis, i.e., end-to-end response times are calculated over node and communication device boundaries.

The precedence relation i modeled by so called *dynamic offset* where the offset for a task represent the earliest possible release of a task based on how early the chain of preceding tasks are able to finish. The jitter term represent the latest possible release time, denoting the time instant where the chain of preceding tasks are able to finish, i.e., the response time of the immediate predecessor of the task at hand. An example transaction with 3 tasks can be seen in Fig. 3.

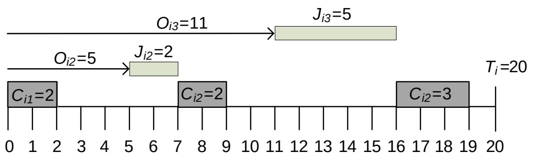

Fig. 3. Precedence example transaction Γ_i

Offset and jitter for the first task, τ_{i1}, is obviously 0 since it is the first task to be executed when the corresponding event arrives. The second task, τ_{i2}, however, is activated when τ_{i1} completes. Here we assume that τ_{i1} will finish no sooner than 5 time units after the event arrival, hence an offset of 5 for τ_{i2}. The latest time τ_{i1} will finish is 7 which means that τ_{i2} will have a release jitter of 7-5=2. Analogous reasoning applies to τ_{i3}. The three tasks in the picture are placed as if all suffer their worst case jitter delays.

Palencia Gutiérrez and González Harbour increased the precision of this analysis by taking impossible combinations into account depending on the priority assignment patterns in [20]. Ola Redell extended the analysis also to handle tree-shaped transactions in [22], as opposed to linear transactions described above.

Hybrid static and dynamic scheduling: In [17] we applied the task model with offset in a system where tasks are scheduled by both static (static schedule) and dynamic (fixed priorities) scheduling policies. There the static schedule is represented by a transaction where offsets represents the time instant where tasks are released according to the static schedule. The jitter term does not come into play for statically scheduled tasks since precedence relations are taken into account by time separation, achieved by the static scheduler. In this work we did a case study of a real example and showed how this system could benefit from hybrid scheduled system. Furthermore, this system model is is commercially available and is supported by the system development tools provided by Arcticus Systems [2].

In conclusion, we see that the task model with offsets has been applied in realistic system contexts and thus has a potential to be adopted by industry provided the remaining deficiencies could be alleviated.

3. Timing models reflecting real systems

When studying the extensive literature on scheduling analysis, it is striking that the only means of interaction between tasks that can be explicitly modeled is synchronization on shared objects using semaphores (or similar simple blocking mechanism). Naturally, this restriction is a large hurdle for any real-time engineer wishing to perform schedulability analysis of systems using the rich set of interaction mechanisms available in contemporary real-time operating-systems.

Traditionally, the only exiting model for task-interaction is blocking on shard resource. This model allows a task to lock a resource for exclusive use for a finite amount of time. The resource would typically be some data-structure shared between a set of tasks. If the operating system manages the locking in clever ways, the amount of time a resource is locked and the effect on tasks trying to lock a busy resource is finite and predictable. Protocols that have this desirable predictable performance include Priority Ceiling Inheritance Protocol, the Immediate Inheritance Protocol, and the Stack Resource Policy [6]. Any serious real-time operating-system should include at least one of these protocols. For these protocols a *blocking factor* (B_{ij} in section 2.1) can be calculated, and subsequently schedulability analysis can be performed.

However, programmers would not, given the choice, restrict themselves to only use such simple task-interaction mechanisms. In fact, programmers expect, and are offered by real-time operating-systems, more advanced interaction mechanisms. Commonly used mechanisms include message passing, event sending, client-server operations, synchronous and asynchronous remote-procedure calls, barrier synchronization, etc. Today, programmers using these mechanisms are basically at lost should they try to perform schedulability analysis of their system.

The task model with offsets (see section 2.1) provides some rudimentary support to model some of the behaviors that occur when using the above interaction mechanisms. The model can be used to express delays and precedence order between different activities (tasks or parts of tasks) in the system. However, how to extract an offset-based model from the code is far from obvious. Also, even if e.g. a client-server call can be modeled by a ordered sequence of executions of task fragments, it may yield quite pessimistic results since we abstract away the particular semantics of the client-server pattern.

So, even if many of the complex execution patterns that exist in industrial real-time systems could be modeled in the task model with offsets we see an immediate need to extend existing schedulability theory with to support at least the following two interaction mechanisms:

- Message passing – Message passing can be used to realize simple precendence relations as expressed in the task model with offsets. However, also more complex interaction patterns may occur in message passing systems. Specifically, a task may block its execution when trying to access an empty message-box. Also, a message-box may contain a queue of messages; representing a backlog of execution to be performed by the task.

  Thus, in order to faithfully model the execution patterns in message-passing systems we need to model the messages queued in the message-boxes. Empty boxes means that message readers will block their execution

and message boxes with more than one message queued will represent a queued execution demand. Analysis techniques to predict queuing patterns, and thus queue-lengths of message boxes are needed.
Message-passing can be used to realize communication of any arity (i.e. 1-to-1, 1-to-many, many-to-1, and many-to-many). Thus, we need to develop models to represent these communication patterns and corresponding analysis techniques that allow these complex relations between tasks.
Exiting techniques to express complex triggering-patterns of tasks include event algebra [7] and timed automata [1]. Combined with the efficient calculation methods provided by RTA, such advanced modeling technique could be used to allow scheduling analysis of large and complex systems using message passing.

- Client-server - Client-server is one of the most common interaction patterns in non real-time code. However, in real-time systems this interaction pattern has limited use. The major reason for this is the inherent unpredictability of the timing behavior of client-server call. Since the timing of client requests is difficult to predict, the resulting load on the server is highly unpredictable; and server response-time is highly dependent on the current load.
  However, in a real-time system, the client-tasks will be real-time tasks with a predictable behavior. Hence, it should be possible to predict the load on the server and, then, also the server response-times. In order to facilitate such analysis we need to device models that allow server-calls to be characterized and their timing to be expressed. We also need models for servers and the services provided by them. These models need to be able to express timing properties for different calls and services. From those timing properties the load on, and response-times from, servers could be predicted.
  Even though the general notion of a server that can perform any tasks on behalf of the caller and even perform synchronization between multiple callers is both useful and unpredictable, it is likely that simpler server-models would also be useful. A simpler model could for instance restrict the server to execute in the thread of the caller - thus simplifying analysis and increasing predictability. One important research topic is to identify the properties a server should have in order to have a predictable behavior. Servers with these properties can then be used in timing-critical real-time systems and analyzed by next generation of schedulability analysis.

4. High precision RTA

Large complex systems that has evolved over several technology shifts and functional upgrades does not lend themselves to academic timing models and analysis of their temporal behavior. One of the main reasons is that the models makes too pessimistic and simplified assumptions about the overall system, often by taking local information and approximate the system behavior with local information. Wall *et al.* recognize that traditional real-time analysis models, such as RTA, are not applicable for large and complex real-time systems [32]. Thus, modeling real systems with current task models and techniques will many times yield the resulting system unschedulabe. However, often the system is well tested and works in practice so there has to be sources of analytical pessimism. We will in this paper identify and discuss two such sources :

- Maximization of local WCET estimation. Often there exist correlations between different tasks execution times in such a way that it is impossible that all tasks in the system experience their worst case execution time simultaneously.
- The current modeling technique to handle precedence relations will yield in pessimistic response time result in very much the same way. By assigning each task local information of release jitter the system wide information about precedence chains is lost.

Furthermore, we believe there might be several other sources of pessimism that exhibits the same pattern of approximating system wide behavior with local information. If such sources could be identified, expressed and analyzed the resulting precision on response times could drastically be reduced. However, we also recognize that there will always be a trade off and a balance between complexity (timing and spatial) on one hand and expressiveness and tightness of RTA on the other hand. In order for RTA to be successfully applied in industrial context one has to find a good and acceptable balance between these two.

4.1. Correlated WCETs

An assumption that all schedulability analysis techniques for hard real-time systems make is that the worst case execution time (WCET) is identified for every task in the system. We will not discuss the open research problem of finding this information [9]. Furthermore, it is assumed that the this execution time of every task is independent of other tasks in the system. However, this assumption is not very realistic since many tasks collaborate and also have data dependencies among them. This will result in overestimated response times.

For following discussion refer to Fig. 4. Any systems interesting temporal end-to-end behavior can be reduced to a transactions. A transaction consists of a set ordered tasks where the first task is activated by an outside event (or clock) and the last task in the set produces some output to the environment (or some internal critical event). With current models each task is modeled with an attribute

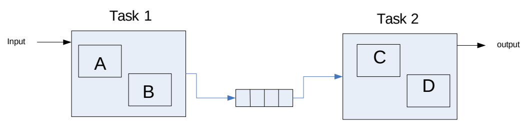

Fig. 4. Correlated WCETs

WCET. Consider the example above where we have two tasks, τ_2 and τ_2, with distinct functionality A and B for τ_1 and C and D for τ_2. The worst case response time for the system (if there are no higher priority tasks in the system) would be $\max(WCET(A), WCET(B)) + \max(WCET(C), WCET(D))$. Assume further that there is a correlation between A and C and B and D, i.e., if A is executed in τ_1 then C is executed in τ_2, and similarly with B and D. If such information could be found and expressed in the timing model the worst case response time would become $\max(WCET(A+C), WCET(B+D))$. That is, we move the maximization from task level to the transaction level, i.e., we are able to express the worst case execution behavior among several tasks, not just locally for every task. This is analogous to calculate with floating point numbers, where task level maximization corresponds to doing round up of each value before performing calculations, whereas transaction level uses the more accurate floating point values in the calculations and rounds them up at the end.

The scientific community should investigate and analyze real industrial systems to find out what kind of dependencies does exist. How can they be expressed, modeled and analyzed? Traditionally the research community on WCET and RTA has been quite independent and have attacked different problems. However, the proposed research direction the WCET community, concerned with local task information, and the RTA community, concerned with global end-to-end response times, should bring the two fields closer together since the research lies in between the two areas. Furthermore, the two different research disciplines should be able to learn and benefit from each others achievements.

4.2. Pessimistic modeling of precedence relations

Another problem we have come across, considering dependencies among tasks, is that modeling precedence relations can sometimes result in pessimistic analysis results. The problem is that precedence relations are modeled with jitter and in the RTA each task has its own local jitter without preserving information about precedence chains. Consider the following transaction with three tasks, each one having a precedence relation to the previous one:

$$\Gamma_i := \langle \{\tau_{i1}, \tau_{i2}, \tau_{i2}\}, T_i = 100 \rangle$$
$$C_{i1} = C_{i2} = C_{i3} = 20$$
$$P_{i1} > P_{i2} > P_{i3}$$

Furthermore, consider a low priority task with $C_{k1} = 30$, for which we are interested to calculate the worst case response time, assuming there are no more tasks in the system.

Assuming that best case execution time approaches zero the jitter for a task is equal to the response time of its predecessor:

$$R_{i1} = 20 \Rightarrow J_{i2} = 20 \Rightarrow$$
$$R_{i2} = 40 \Rightarrow J_{i3} = 40$$

The release jitter term means that a task that is released periodically may some times be delayed at most with its jitter term. This has an impact when considering the response time of τ_{k1} where the interference of each higher priority task is considered separately. Therefore, the worst case interference τ_{i2} imposes τ_{k1} is when it first suffers its worst case jitter and subsequent releases occurs with no jitter. The resulting scenario is depicted in Fig. 5. The worst

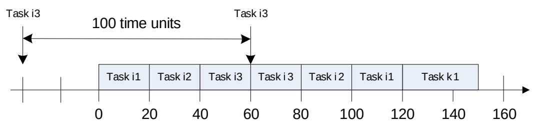

Fig. 5. Correlated WCETs

case interference occurs when a task is released, experiences its worst case release jitter, coincides with the release of the task under analysis, and finally released again after period time units (100 in this example) has elapsed from previous release. In the figure this fact is highlighted for τ_{i3}. Similar reasoning applies to τ_{i2}. Hence the response time of τ_{k1} becomes 150 time units. The reason for this is that it is assumed that all tasks experience their worst case jitter simultaneously and independently. However, in order for to really experience its worst case release jitter τ_{i1} and τ_{i3} would have to execute for almost zero time units and hence τ_{k1} can not experience the worst case interference from them. In essence, τ_{k1} can not experience both the WCET of all three tasks and at the same time as they all experience their maximum release jitter delay. In reality τ_{k1} can only be interfered by each task at most once and hence the worst case response time in reality can not exceed 90 time units

In order for RTA to be industrially applicable, this pessimism must clearly be addressed since precedence relations is common type of inter-task dependency and are widely used in industrial systems.

5. Conclusion

The scientific community has produces an overwhelming amount of research result on schedulability and temporal analysis of embedded real-time systems. However very little of these result have gained industrial acceptance, rather it is easier to find documents of negative results of trying to use such results in industrial systems .

This paper recognizes several shortcomings of these extensive results and proposes some research directions, in the area of Response Time Analysis (RTA), in order to gain industrial strength timing analysis models that would be accepted by the industrial community. RTA has over many years matured to a degree where models can express complex system behaviors and analysis results are relatively tight with respect to real system behavior. However there is still work to be done.

The core problem is that the scientific community uses too simplistic or research oriented timing models that does not reflect real industrial systems to an acceptable degree. We argue that there are two main reasons, and thus needs further research efforts, RTA has not reached the industrial community:

1) Lack of capabilities in the models used by schedulability analysis techniques. Industrial real-time systems use a variety of constructs to interact between tasks. Existing schedulability analysis only allow interaction patterns that use simple blocking on shared resources and precedence constraints between tasks. We identified message-passing and client-server as two key mechanisms that are commonly used in real systems. Thus, RTA needs to be extended to allow modeling and analysis of tasks using these interaction mechanisms.
2) Lack of precision in analysis techniques. Although timing models can handle many real scenarios and complex inter-task dependencies, these models are too simplistic in the sense that local information on parts of the system is used to approximate global behavior system behavior. This leads to pessimistic and imprecise analysis results. In order to increase the precision one must find ways to find, express, and utilize inter task dependencies that gives more accurate information on bigger parts of the system and not just on task level.

To conclude, we believe that RTA has the potential to be a useful engineering tool for developers of embedded real-time systems by providing both formal guarantees on temporal behavior leading to less testing and to ease lengthy certification processes for safety-critical applications as well as providing an understanding of the systems overall temporal behavior.

6. Acknowledgments

This work was partially supported by the Swedish Foundation for Strategic Research via the strategic research centre PROGRESS and by the Knowledge Foundation within the project MultEx.

References

[1] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. Times: A Tool for Schedulability Analysis and Code Generation of Real-Time Systems. In *International Workshop on Formal Modeling and Analysis of Timed Systems*, 2003.

[2] Arcticus Systems Web-Page. http://www.arcticus-systems.com.

[3] N.C. Audsley, A. Burns, R.I. Davis, K. Tindell, and A.J. Wellings. Fixed Priority Pre-Emptive Scheduling: An Historical Perspective. *Real-Time Systems*, 8(2/3):173–198, 1995.

[4] N.C. Audsley, A. Burns, K. Tindell, M.F. Richardson, and A.J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[5] A. Burns, K. Tindell, and A Wellings. Effective Analysis for Engineering Real-Time Fixed Priority Schedulers. *IEEE Transactions on Software Engineering*, 22(5):475–480, May 1995.

[6] G.C. Buttazzo. *Hard Real-Time Computing Systems*. Kluwer Academic Publishers, 1997. ISBN 0-7923-9994-3.

[7] Jan Carlson and Björn Lisper. An Event Detection Algebra for Reactive Systems. In *Proceedings of the 4th ACM International Conference on Embedded Software (EMSOFT)*, 2004.

[8] International Electrotechnical Commission. IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems.

[9] J. Engblom, A. Ermedahl, M. Nolin, J. Gustafsson, and H. Hansson. Worst-Case Execution-Time Analysis for Embedded Real-Time Systems. *International Journal on Software Tools for Technology Transfer*, 4(4):437–455, October 2003.

[10] A. Ermedahl, H. Hansson, and M. Sjödin. Response-Time Guarantees in ATM Networks. In *Proc. 18th IEEE Real-Time Systems Symposium (RTSS)*, pages 274–284. IEEE Computer Society Press, December 1997.

[11] M.H. Klein et al. A Practitioners Handbook for RMA.

[12] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, 1986.

[13] D.I. Katcher, H. Arakawa, and J.K. Strosnider. Engineering and analysis of fixed priority schedulers. *IEEE Transactions on Software Engineering*, 19(9):920–934, September 1993.

[14] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proc. 11th IEEE Real-Time Systems Symposium (RTSS)*, pages 201–212, December 1990.

[15] R. Lencevicius and A. Ran. Can Fixed Priority Scheduling Work in Practice? In *Proc. 24th IEEE Real-Time Systems Symposium (RTSS)*, page 358, December 2003.

[16] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

[17] J. Mäki-Turja, K. Hänninen, and M. Nolin. Efficient Development of Real-Time Systems Using Hybrid Scheduling. In *International conference on Embedded Systems and Applications (ESA)*, June 2005.

[18] J. Mäki-Turja and M. Nolin. Efficient implementation of tight response-times for tasks with offsets. *Journal of Real-Time Systems*, February 2008.

[19] J.C. Palencia Gutiérrez and M. Gonzáles Harbour. Schedulability Analysis for Tasks with Static and Dynamic Offsets. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS)*, December 1998.

[20] J.C. Palencia Gutiérrez and M. Gonzáles Harbour. Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems. In *Proc. 20th IEEE Real-Time Systems Symposium (RTSS)*, pages 328–339, December 1999.

[21] S. Punnekkat. *Schedulability Analysis for Fault Tolerant Real-time Systems*. PhD thesis, University of York, June 1997.

[22] O. Redell. Analysis of tree-shaped transactions in distributed real time systems. In *Proc. of the 16th Euromicro Conference on Real-Time Systems*, June 2004.

[23] L. Sha, T. Abdelzaher, K-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok. Real Time Scheduling Theory: A Historical Perspective. *Real-Time Systems*, 28(2/3):101–155, 2004.

[24] L. Sha, R. Rajkumar, and J.P. Lehoczky. Task scheduling in distributed real-time systems. In *IEEE Industrial Electronics Conference*, 1987.

[25] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority Inheritance Protocols: an Approach to Real Time Synchronization . *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

[26] H. Thane and H. Hansson. Towards systematic testing of distributed real-time systems. In *Proc. 20th IEEE Real-Time Systems Symposium (RTSS)*, pages 360–369, December 1999.

[27] K. Tindell. An extendible approach for analyzing fixed priority hard real-time tasks. Technical Report YCS189, Dept. of Computer Science, University of York, England, 1992.

[28] K. Tindell. Using Offset Information to Analyse Static Priority Pre-emptively Scheduled Task Sets. Technical Report YCS-182, Dept. of Computer Science, University of York, England, 1992.

[29] K. Tindell and A. Burns. Fixed Priority Scheduling of Hard Real-Time Multimedia Disk Traffic. *The Computer Journal*, 37(8):691–697, 1994.

[30] K. Tindell and J. Clark. Holistic Schedulability Analysis For Distributed Hard Real-Time Systems. Technical Report YCS197, Real-Time Systems Research Group, Department of Computer Science, University of York, November 1994. URL ftp://ftp.cs.york.ac.uk/-pub/realtime/papers/YCS197.ps.Z.

[31] K. Tindell, H. Hansson, and A. Wellings. Analysing Real-Time Communications: Controller Area Network (CAN). In *Proc. 15th IEEE Real-Time Systems Symposium (RTSS)*, pages 259–263. IEEE, IEEE Computer Society Press, December 1994.

[32] A. Wall, J. Andersson, and C. Norström. Probabilistic Simulation-based Analysis of Complex Real-Times Systems. In *6th IEEE International Symposium on Object-oriented Real-time distributed Computing*, Hakodate, Hokkaido, Japan, May 2003.