Improving Temporal Behavior with Graphical Method in Real Time Systems

Francis Cottet, Laurent David

IFAC Workshop On Real Time Programming, Palma de Mallorca, Spain, April, 2000, pp. 79-84

IMPROVING TEMPORAL BEHAVIOR WITH GRAPHICAL METHOD IN REAL-TIME SYSTEMS

F. Cottet and L. David

LISI / ENSMA - B.P. 40109 - 86961 FUTUROSCOPE Cedex - FRANCE
Ph. (33)5-49-49-80-52 / Fax (33)5-49-49-80-64 / e-mail : {cottet, david}@ensma.fr

Abstract : The purpose of this work is to provide to the designer a graphical method for taking design decisions for enhancing the temporal performance of the application in terms of fault tolerance, task response time, etc. We have introduced a graphical representation of the schedulability analysis based on a classical temporal model in a given scheduling environment. This methodology is extremely useful for supporting both control applications and multimedia systems, in which the execution times or rates of some computational activities can vary significantly, and for which the timing fault tolerance must be evaluated in order to prevent failures.

Keywords : validation of real-time systems, timing analysis, fault-tolerant software, scheduling algorithms.

1. INTRODUCTION

A real-time system is, by definition, a system that must satisfy explicit response-time constraints, or risk severe consequences including failure (Laplante, 1993). Since most of the mission-critical and human-critical systems operate in real-time, these also require a high degree of dependability. It can then be sustained that two of the most important characteristics of real-time systems are the capability to fulfill strict timing requirements and to guard against imperfect execution environments.

In real-time systems, the time is a main point, unlike the definition for non-real-time systems, where time is only a performance matter. So an appropriate approach is a temporal analysis and validation of the system before its execution in order to guarantee the timing constraints of the application. One method, increasingly used, is based on schedulability analysis. Various methodologies are founded either on specific real time languages (Stoyenko, 1992), or on a temporal modeling of software written with a classical high level language associated with a real-time kernel (Cottet et al., 1994; Zhou et al., 1998), or on formal methods that lead to exhaustive computation of the scheduled task execution sequences (Geniet et al., 1996).

This operational validation by schedulability analysis forces the mathematical characterization of the task set according to a precise temporal model. In the simplest widely accepted model, a task T_i, assumed to be periodic and independent, is represented by the quadruple (r_i, C_i, D_i, P_i) where r_i is the first request of the execution of the task, C_i is the computation time, D_i is the deadline and P_i is the period. We have the basic obvious conditions (1) ($C_i \leq D_i \leq P_i$) where $D_i \leq P_i$ means that the task deadline is lower than the next execution request (the task must be completed before a new invocation) and $C_i \leq D_i$ the necessary condition for the execution.

Three of these parameters (r_i, D_i, P_i) are elicited from the timing requirements giving the temporal characteristics of the input/output operations and the process behavior. The temporal parameter C_i corresponds to the execution time of each task in the intended hardware and software execution environment. It can be evaluated by different ways : either from an analysis of the program code (Puschner et al., 1993) or from a direct measurement of execution time (Babau et al., 1995). It is usually fixed and depends only on the algorithmic implementation in a given hardware context. It is possible to estimate a variation range from a minimum run-time $C_{i,0}$ to the worst case execution

time $C_{i,m}$ of a task. The others parameters can be adjusted by the designer to obtain a system that matches as closely as possible the timing constraints of the application. Particularly, this adjustment of the deadline and the period of the task is extremely important because, in relation to the formal scheduling algorithms such as Rate Monotonic and Earliest Deadline (Liu and layland, 1973), it corresponds to a priority mapping that takes into account the importance of tasks.

The purpose of this work is to provide to the designer a graphical method of task temporal parameter adjustment in order to greatly improve the temporal behavior of a hard real-time system. This approach is particularly useful to enhance the performance of the application in terms of quality of temporal service (task response time, processor idle time, number of processor context switches, etc.). Moreover this graphical method can be used to provide tolerance of timing faults caused by the effect of the parameter variation as the period jitters or the computational time uncertainty. It provides an approach to reliable real-time systems. The main investigations for improving fault tolerance of hard real-time systems were realized by modifying task model (Kim et al., 1998). The paper is organized as follows. In section 2, we present the graphical model of a task. In particular, we choose to describe more precisely the plane "deadline versus period". In section 3, we introduce the schedulability condition in the graphical representation. A simple example ends this section. Section 4 presents the graphical analysis of the plane "execution duration versus period". A speediness increase of this methodology and a real industrial example are presented in the section 5. Conclusion are given in section 6.

2. GRAPHICAL MODELING OF TASKS

2.1 *Scope and assumptions*

This work is focused on one-processor and preemptive real-time systems. Concerning the tasks, we simply identify two different kinds of activation signals. The first concerns the periodic signals, characterized by a period P, which activate hardware periodic tasks. The second class contains the non-periodic signals that occur at arbitrary times without a fixed time interval. In this situation, we need a classical specification that is given by the minimum interval d_{min} between two occurrences of such signals

For this latter case, we have adopted the simplest approach very practical in real-world applications : the immediate periodic server (the server task services any pending non-periodic requests, if no non-periodic requests are pending, the server suspends itself until its next period and the released time is instead used for periodic tasks) (Homayoun and Ramanritham, 1994; Spuri and Buttazzo, 1996; Buttazzo, 1997). So from now on, we will consider that a periodic server is created and dedicated to each non-periodic request. Therefore we assume that a real-time software is depicted by periodic task set.

2.2 *A three-dimensional general model of a task*

Assuming that each task has a known initial invocation r_i, the key point is to skillfully determine the other temporal parameters (C_i, D_i, P_i) according to the non-functional requirements such as the timing properties of the application.
First, the parameters of the task temporal model (C_i, D_i, P_i), as previously presented, must satisfy the relations (1). Therefore, in a set of axes (run-time, deadline, period), the three-dimensional graphical representation of the volume corresponding to the allowed values of task parameters has a trihedral shape. If we draw the projections of this sphere on the different planes, we obtain the areas of the allowed values of task attributes : plane projection (period versus run-time), front elevation (deadline versus period) and side elevation (deadline versus run-time). All theses surfaces come in the form of an opened wedge. The vertices of these acute angle wedges correspond to the equality of the parameters: $C_i = R_i = P_i = C_{i,0}$. So, if we assume that the computational time varies in the range $[C_{i,0}, C_{i,m}]$, then, in the deadline versus period plane for example, the allowed zone changes as described on the figure 1. The more the run-time increases, the more the possible surface of task parameter values shrinks.

We confine our attention here to two of these projections that seem quite interesting for our purpose of temporal analysis of task behavior : deadline-period and period-execution time co-ordinate planes. We first assume that each task has a known worst case upper bound to the execution time that can be computed ($C_i=C_{i,m}$). So the free other parameters (D_i, P_i) can be studied according to the timing requirements of the task itself, derived from the global timing properties of the application. The effect of relaxing the first assumption ($C_i=C_{i,m}$=constant) will be discussed in section 4.

2.3 *Graphical temporal modeling of periodic tasks*

Up to now, the task was considered as a theoretical model of periodic task without taking into account some timing requirements of the application.

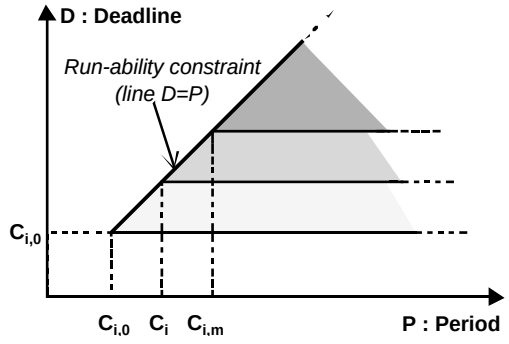

Fig. 1 : A graphical representation of the possible values of temporal parameters of a task.

From these temporal constraints related to the controlled process, different temporal characteristics of a task can be deduced and imposed ; for example, we can have the following requirements :

• a maximum period $P_{i,max}$ ($P_i \leq P_{i,max}$), corresponding, for instance, to a sampling rate limit,

• a minimum period $P_{i,min}$ ($P_i \geq P_{i,min}$), corresponding to a useless repetition rate (for instance the period of a polling task with a value much greater than the physical signal change rate),

• a maximum deadline $D_{i,max}$ ($D_i \leq D_{i,max}$), in order to get a better response time.

The graphical representation of figure 2 shows the previously determined area of the allowed values of task attributes located inside the shaded wedge, named zone A. On the other hand we obtain a more restricted surface (zone B), included in the first one, that corresponds to the set of potential couples (D_i, P_i) of a given periodic task satisfying its timing constraints.

2.4 - *Temporal modeling of server tasks*

A non-periodic event is assumed to be specified by at least a minimum time interval between two invocations, d_{min}, and if necessary a maximum deadline, D_{ev}. So we have to find the temporal parameters (D_{si}, P_{si}) of the periodic server that offers a correct service to any of those non periodic invocation signals, that is to say : to respond with a delay less than its deadline D_{ev}. As before, we assume that the computation time C_{si} allocated to the immediate server is known. Given that there is of course no synchronization between the periodic server activation and the occurrences of the non periodic events, we are going to consider the worst case. Suppose that the non periodic event occurs just after the release time of the periodic server and that it suspends immediately since there is no pending non-periodic signal. Therefore the non periodic request event will be served during the next period of the server but before the server deadline attached to this new period. Due to that, the temporal parameters must meet the following condition :

$$P_{si} + D_{si} \leq D_{ev} \leq d_{min} \qquad (2)$$

This condition, involving P_{si} and D_{si}, offers a certain choice of the periodic server parameters : either a long period server with a short deadline that strongly constraints the schedulability of the task set or a short period with a deadline equal to the period that is prejudicial to the processor load.

3 - SCHEDULABILITY HANDLING IN THIS GRAPHICAL MODELING

The temporal model representations so far show the allowed main temporal parameters of the task with the effect of the timing requirements, but without taking into account the influence of other tasks, or, in others words, the concurrent execution context.

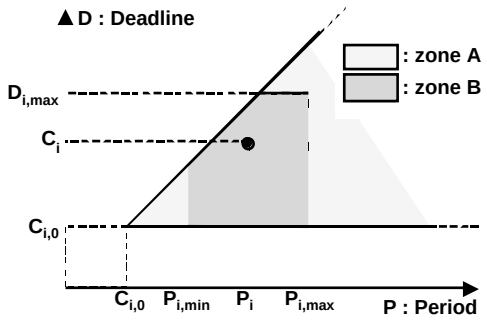

Fig. 2 : A graphical representation of the possible values of temporal parameters (D_i, P_i) for periodic tasks corresponding to the temporal requirements.

The aim of this work is to provide a methodology of task set analysis for determining the well adapted temporal parameters. The best fitting parameters are considered with respect to the global timing requirements of the application. This property requires a framework that incorporates the execution environment, especially the scheduling algorithm. We have first chosen a static scheduling algorithm (the priorities are not changed once they are assigned to the tasks) namely the Rate Monotonic (RM) algorithm. This scheduling algorithm is a simple rule that assigns priorities to tasks according to their request rates. Specifically, tasks with shorter periods will have higher priorities.

Let us consider a real-time system with k tasks (basic periodic task and/or server task). The worst-case computation time C_i of each task T_i is assumed to be known in a deterministic manner. Our approach to fixing temporal task parameters of the task set works according to an iterative process. At a given step, we assume that the parameters of k-1 tasks have been determined, that is to say, their point (D_i, P_i) or (D_{si}, P_{si}) has been chosen in the allowed areas. From these assumptions, we can build the graphical representation (D_k, P_k) of the task k with its proper restricted surface and the boundaries representing the interference of the concurrent execution context caused by the k-1 existing tasks. This methodology can be seen according to two different points of view. Either, in a design process, the k-th task must be added to the others or, in a re-design or analysis process, the k-th task is one among the others whose temporal characteristics one wants to improve.

The first obvious bound is given by the maximum processor utilization factor, that is to say 1 in a one-processor context. Therefore, P_k has a minimum value P_{lim} which is given by :

$$P_k \geq P_{lim} = C_k / (1 - \sum_{i=1}^{k-1} C_i / P_i \qquad (3)$$

But the main limitation that we have to take into account is the schedulability constraint. For a RM scheduling and a task set where each task must be completed before the new invocation, a whole task set is schedulable if [Liu and Layland, 1973] :

$$\sum_{i=1}^{k-1} \frac{C_i}{P_i} + \frac{C_k}{P_k} \le k\left[2^{1/k} - 1\right] \qquad (4)$$

In this particular assumption (run-ability constraints for all the tasks) the preceding condition gives us a minimum period $P_{LL,lim}$ located on the "D = P" line. It should be noted that this condition is sufficient but not necessary. It should be pointed out that these different limitations are often very pessimistic. To improve efficiency of this computer aided design methodology, we have to do a simulation of the RM scheduling of the task set. Then the designer can choose the appropriate pair of temporal parameters (see figure 3). This simulation must cover scheduling for a length of time equal to the major cycle corresponding to the least common multiple of the task periods $LCM\{P_i\}$.

In the case of a server task (example of the figure 3), a very important limit can be defined : the maximum response time TR of the task that corresponds to the sum of the deadline and the period. In that case also, this limit is an upper bound, the server task can be replied more quickly to the aperiodic interruption in a specific execution sequence. So, for a given scheduling algorithm, the direct computation of valid sequences permits us to draw the real and complete limits of the allowed zone for a given task in a concurrent scheduling environment. This final zone (C), that is included in the zone B (previously inside the zone A), can be much more restricted than the previous surfaces. The borders of this surface are constituted by of different lines : the D=P line where we find the analytical results for the period, the specification upper value of the deadline, the maximum response line, the minimum deadline that can be irregularly shaped. The final zone C emphasize the capability of the system to tolerate timing fault related to the deadline and the period of this particular task.

First representing the results of an simple example in the D-P plane is slight different from the preceding theoretical graphs because the D and P variables are integers and therefore the D-P plane is discrete. Let the test example be a periodic task set with two tasks characterized by the following temporal parameters : (0, 2, 8, 10) and (0, 2, 8, 8).

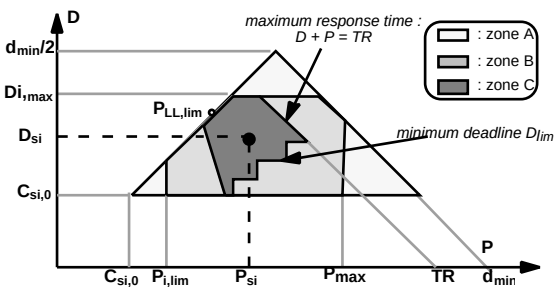

Fig. 3 : The allowed temporal attributes of the periodic server in a multitasking environment.

Assume that we want to add a periodic server task created in order to deal with a non-periodic event, characterized by a needed computation time C_3 of 3 and a minimum occurrence interval d_{min} of 15. We can calculate the minimum period corresponding to the fully used processor according to the equation (3) : P_{lim} = 6. To obtain the complete graph, the RM scheduling has been applied for the predetermined range of P_3. The figure 4 shows the final results with the allowed zone of temporal parameters of this task. The designer can chose among all the points the possible values of D and P. There is no better choice among all these possible parameters. Different criteria can be used to select the task temporal characteristics such as minimizing the processor load : point A(10, 5). Another way is to optimize two criteria like processor utilization and response time. In this case, the point, referenced B(8, 7) on the figure 4, seems to be a very interesting compromise between the highest available period and a correct response time.

In all the above execution simulations, we consider a RM scheduling. Before modifying the temporal parameters of a task, a designer could also want to analyze how a more efficient scheduling policy can widen the valid zone. So we now turn our attention to obtaining valid area corresponding to another scheduling algorithm: Earliest Deadline (ED) (see figure 5). Referring to the previous results about the valid area obtained with a RM scheduling, the use of ED algorithm really offers new choices of task parameters : 23 points instead of 13). For example, with a strongly constraint deadline equal to 3, the period can be varied over a range as wide as 5 : P [7, 12]. For the choice of parameters, our server can be implemented with parameters D=3 and P=12 which unload the processor considerably while keeping the required response time for the aperiodic event.

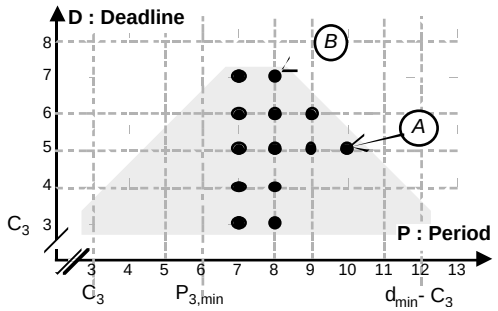

Fig. 4 : Example with a RM scheduling.

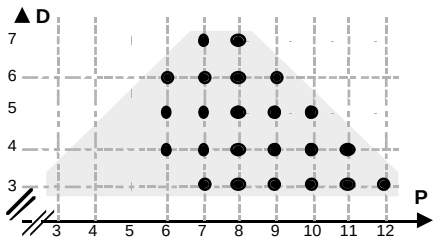

Fig. 5 : Example with ED scheduling.

4 - TEMPORAL ANALYSIS IN THE PERIOD VS RUN-TIME REPRESENTATION

In previous interpretation of the D-P plane mapping, the goal was focused on the evaluation and the improvement of the temporal characteristics of a task set. With a period versus run-time plane, the purpose is to analyze the ability of the system to undergo temporal faults (period jitters, run-time variation) without jeopardizing the schedulability of the whole task set. In the previous case, we assumed that the computational time was constant and equal to a known worst case execution time ($C_i = C_{i,m}$). But, in fact, the maximum value of the run-time of a task is quite difficult to estimate with a high accuracy because of the laborious analysis of the execution paths of the software, the compiler effects, the hardware environment (caching, paging). Another reason to study a task with a variable run-time is that some algorithms, used in task software, contain loops that yield more accurate results with an increasing number of iterations. So, it should be very useful to know the upper limit of the execution time that a task can possess. Assuming that the deadline D_i was set earlier and the possible variation range of the run-time was also determined ($C_i \in [C_{i,0}, C_{i,m}]$), we can construct the valid area of the task. First we plot the opened wedge corresponding to the initial model described in section 2.2. (zone bounded by the two lines : $C=C_{i,0}$ and $P=C$). Then the temporal requirements of the application reduce the allowed parameters. And finally the schedulability analysis gives the final possible temporal characteristics of the task. From that result, it is quite easy to evaluate the possible variation of the execution time at a given period or the inverse. Under this framework, periodic tasks can change their execution rate or their execution times to provide different quality of service (fault tolerant, system load, etc.). This will be developed in details for a real example in the next section.

5 - ANALYZING A REAL EXAMPLE

5.1 - Speediness improvement of this graphical tool

A tool, based on this computer aided design methodology, was developed with a user-friendly interface that assists real-time designers in fixing the parameters of a task set of an application. And we can remark that each point of the graphical representation corresponds to a complete simulation of the application scheduling, even if the point parameters lead to a non-schedulable result. Furthermore this duration of simulation can be extremely large. So, we take care to minimize the number of parameter cases to check by using analytical conditions and properties of scheduling. We want to strongly reduce the simulated area from a number of point that is theoretically proportional to P_{max}^2 in a periodic task case with a maximum period of P_{max}. First, we have established some graphical properties coming from scheduling characteristics. These properties permit us to draw a line of feasible points without any schedulability calculations. For example, in RM scheduling context, we have the following results :

- property 1 : if the schedulability test is positive for the value couple (D_x, P_x), it will be correct for the other value couples such as ($D_x + \alpha$, P_x).
- property 2 : if the schedulability test is positive for the value couple (P_x, C_x), it will be correct for the other value couples such as (P_x, $C_x - \alpha$).
- property 3 : if the schedulability test is positive for the value couple (D_x, P_x), it will be correct for the other value couples such as (D_x, $P_x + \alpha$) if the task keeps the same priority.
- property 4 : if the schedulability test is positive for the value couple (P_x, C_x), it will be correct for the other value couples such as ($P_x + \alpha$, C_x) if the task keeps the same priority.

Using these properties, associated to a dichotomic analysis among the possible couples of task parameters, the number of points to simulate is limited to number of point proportional to $\log(P_{max})\ P_{max}$ in the RM scheduling context and to $\log(P_{max})+\log(P_{max})$ with DM scheduling.

5.2 – An industrial case study

To illustrate the methodology presented in previous sections and to experiment the graphical approach, we analyze a realistic example. This industrial case study is based on an aluminum cold rolling mill application. The studied application concerns the software dedicated to the real-time check of the product quality (specially the thickness regularity). The control process software is supported and managed by another computer. The system architecture is based on both Motorola 2600 board and LynxOS™ real-time operating system. There are ten tasks presented in the Table 1 where the temporal parameters are given in milliseconds. The task set parameters used for the example denote the actual or nominal characteristics (period and execution time). The processor utilization is 0.84. The schedulability must be studied over a duration of 200 s corresponding to the LCM of task periods. The processor is idle in the remaining time 32 s.

Table 1 : A real-time application with ten tasks

Task	C (ms)	D (s)	P (s)
T_1	250	1	2
T_2	500	4	4
T_3	250	4	4
T_4	500	4	4
T_5	250	4	4
T_6	500	3	8
T_7	1,250	10	20
T_8	2,500	50	100
T_9	12,500	200	200
T_{10}	12,500	200	200

We confine our attention here to the data processing task 3 that is constrained to be in the period range [1s, 4s], the actual period being 4 s. On the other

hand, the computational time of this task can be varied from an execution to another because of the calculations depending of the data number.

Using the presented methodology, we obtain the graphical result of the schedulability domains with RM and DM scheduling algorithms (see figure 6). For the period 2 s, the computational time can vary from 250 ms to 370 ms for a RM priority assignment and from 250 ms to 500 ms for a DM priority mapping. The major conclusion in this case is that the timing fault tolerance of the system is increased by a factor 2 concerning this task 3. If the task period can be set to the upper value of the specified period range ($P_{3,max}$ = 4 s), the execution time is in the range [250 ms, 750 ms] for RM scheduling and [250 ms, 1.1 s] for DM scheduling. In this latter case ($P_{3,max}$ = 4 s, DM scheduling), the application is schedulable for a processor utilization varying from 0.78 to close to 1 due to the task 3 execution time variation.

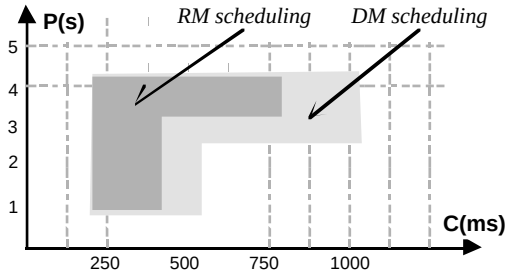

Fig. 6 : The allowed temporal parameters of the task 3 of the example described in Table 1.

6 - CONCLUSION

In this paper, we have introduced a graphical representation for tasks based on a classical temporal model. The elaboration of the valid zone for a given task is computed in three steps which take into account the intrinsic characteristics of the periodic task, the timing requirements of the application for this particular task and finally the schedulability of the task set. These successive surfaces are of course smaller and smaller. At the end, the user has to take design decisions and to chose the right parameters of the task in the way to improve the temporal performances of the task itself and more globally of the application.

With a representation in deadline-period co-ordinate plane, this approach is particularly useful in situations in which the designer has to determine parameters of a task set, either to add a task to a set of tasks with already fixed parameters or to modify the task parameters in order to analyze and improve some temporal characteristics (response time, processor load, …). With a specific representation (period versus run-time plane), the purpose is to analyze the ability of the system to tolerate temporal faults (period jitters, run-time variation, …) without risking the schedulability of the whole task set. Moreover, we have shown how this graphical model permits us to compare the different scheduling algorithms : RM, DM and ED.

REFERENCES

Babau,J.P. ,S. Gérard and F. Cottet "Measurement of task execution durations in a real-time environment", *Proceedings of the 4th Real-Time Systems*, (Paris, France, January 10-12,1995).

Buttazzo, G.C. “Hard Real-Time Computing Systems, Predictable Scheduling Algorithms and Applications”, ed. Kluwer Academic Publishers, 1997.

Cottet F. and J.P. Babau "Off-Line Temporal Analysis of Hard Real-Time Applications", *Proceedings. of the 2nd IEEE Workshop on Real-Time Applications*, (Washington, DC, July 21-22,1994).

Geniet, A., D. Geniet, F. Cottet, “Exhaustive Computation of the Scheduled Task Execution Sequences of a Real-Time Application”, *Proceedings of 4th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems* (Uppsala, Suéde, September 11-13, 1996).

Homayoun, N., P. Ramamritham "Dynamic Priority Scheduling of Periodic and Aperiodic Tasks in Hard Real-Time Systems", *The Journal of Real-Time Systems*, 6, 207-232 (1994).

Kim, H., A.L. White, K.G. Shin, “Reliability Modeling of Hard Real-Time Systems”, *Proceedings of the 28th International Symposium on Fault-Tolerant* (June 1998, Munich, D)

Laplante, P., ”Real-Time Systems Design and Analysis – an Engineer’s Hand Book”, Ed. *IEEE Computer Society Press*, 1993.

Liu C.L. and J.W. Layland "Scheduling algorithms for multiprogramming in a hard real-time environment", *Journal of ACM*, 20 (1), 46-61 (1973).

Puschner P. and A. Schedl "A Tool for the Computation of Worst Case Tool Execution Times", *Proceedings. of the 5th Euromicro Workshop on Real-Time Systems* (Oulu, Finlande), 224-229 (1993).

Spuri M. and G.C. Buttazzo "Scheduling aperiodic tasks in dynamic priority systems" *Journal of Real-Time Systems*, December 1996.

Stoyenko, A. "The evolution and State-of-the-Art of Real-Time Languages", *The Journal. of Systems and Software*, 61-84 (April 1992).

Zhou, L., K.G. Shin, E.A. Rundensteiner “rate-Monotonic Scheduling in Presence of Timing Unpredictability” Proceedins of the 4th IEEE Real-Time Rechnology and Applications Symposium, (Denver, June 3-4) 1998.