Formal evaluation of Quality of Service for data acquisition systems

Belgacem Ben Hedia, Fabrice Jumel, Jean-Philippe Babau

CITI - INSA Lyon - CPE
F69621 Villeurbanne Cedex - FRANCE
e_mail : {belgacem.ben-hedia; fabrice.jumel; jean-philippe.babau }@insa-lyon.fr

Abstract

In the field of real-time control applications, validation relies on a precise knowledge of the temporal characteristics of the used data such as delays and loss rates. These data are provided by a dedicated software called the driver. Consequently, it is necessary to evaluate the impact of the driver on the QoS (Quality of Service) of the data. This work proposes a formal model of data drivers based on communicating timed automata and shows how parameters of the driver impact the provided QoS of data.

1. Introduction

Computers are more and more used for process control systems (automotive, railway, industrial processes). These computers implement control laws using data coming from sensors and producing commands to actuators. Due to interactions and dynamic evolution of controlled process, the control law implementation needs to satisfy some critical QoS constraints (data arrival laws, data lost rate, time interval between command and data update).

Nowadays, these systems are more and more complex and the reuse of software components may reduce the development costs. In process control system domain, reusable components are classically execution services provided by a real time operating system respecting standards such as OSEK [5] or POSIX [16]; remote communication services (FT layer of TTA [17]); and device drivers. The purpose of the device driver is to provide an interface between the physical sensors and the applications. In this work, we consider the data drivers, that is to say the data acquisition part of process control system [14].

Due to QoS constraints, it is necessary to produce a guaranteed evaluation of QoS characteristics of drivers. In this work, we consider three classical QoS characteristics for process control system: maximum and minimum delays, maximum loss rates and maximum consecutive losses.

The most common techniques to obtain maximum delays are based on tasks and messages scheduling analysis [1, 19, 20, 23]. These techniques consider simple architectures, generally static, and give unrealistic bounds. In order to limit this problem, many works are using techniques based on exhaustive modelling of temporal behaviours such as communicating timed automata or hybrid automata [3, 4, 11, 13, 15, 18]. These works make possible to obtain general and realistic information about temporal application behaviour of complex architectures (dynamic behaviours).

This paper aims to evaluate QoS characteristics of the data produced by driver. Because the architecture of data acquisition system depends on many parameters such as buffer size and communication policies and because the considered temporal behaviour is complex (arrivals law from the sensors, loss rate), exhaustive modelling based on communicating timed automata seems more suitable. Among existing formalisms, IF language is a well suited candidate. Indeed, it provides a full tool chain for formal evaluation (LTS generation, abstraction, model checking tool [26]).

The rest of this paper is organized as follows. The next section presents the data acquisition systems and the desired QoS characteristics. The third section presents the modelling strategy. Then, the last section illustrates the approach by analysing the impact of several parameters on the QoS characteristics. We finally conclude the paper and give some perspectives.

2. Context

2.1 Data acquisition system

A data acquisition system (c.f. figure 1) is made of four different elements:

- **Physical sensor:** a physical sensor converts information (temperature, speed) coming from the environment to numerical data.
- **Communication interface:** a communication interface connects a sensor to the software part of the system (analog-to-digital converter). It recovers data produced by the physical sensor and stores them into a buffer accessible by the driver.
- **Device driver**: a device driver is a dedicated software integrated into the operating system and independent of the application. It abstracts the hardware layer and can be used in various applicative contexts. A device driver can be viewed as a "communication middleware" between the controlled environment and a real time application. It transmits the data, stored into the device buffer, to the application.
- **Real time application**: a real time application reads the data from the device drivers.

We make a distinction between the hardware driver (called communication interface) and the software driver (called device driver).

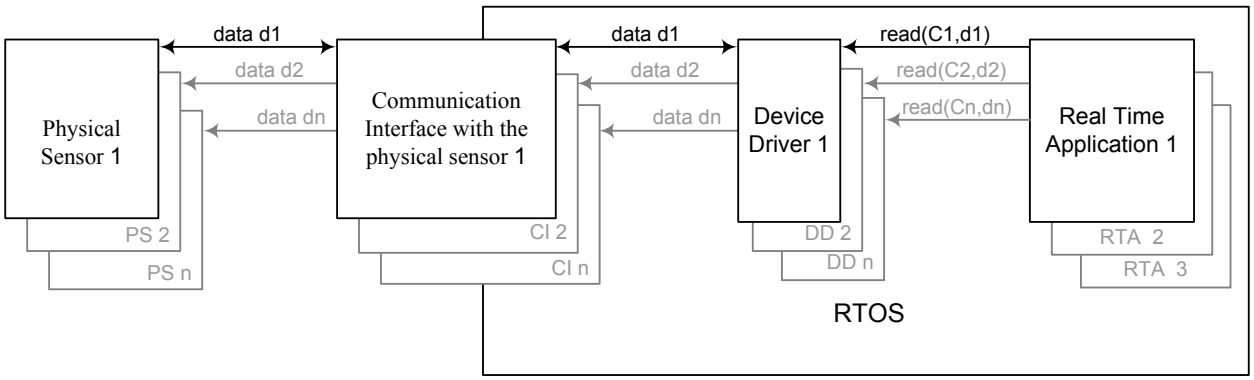


FIG. 1 – Data acquisition system model

2.2 Definition of QoS criteria

Depending on applications, the most relevant QoS criteria can differ [24]. Thereafter, the considered QoS are:

- **Delay**: data used by the application reflect the process at a valid state but at a previous instant. The delay is the interval of time between the instant when data changes on the controlled process and the time when the application read it. We consider only **maximum delay** and **minimum delay**.
- **Loss rate:** some information on controlled process will never arrive to the application and are lost. The loss rate is the percentage of the amount of lost data by the amount of produced data. We consider only the **maximum loss rate**.

2.3 Formal techniques and tools

A large number of works are related to the performance evaluation of the drivers used in large information storage equipments. These approaches are generally based on execution observation techniques (temporal trace analysis) [12, 2], or on simulation of models [11]. The main objective of these works is the delay evaluation. But the used techniques are based on statistical or probabilistic analysis. The obtained results are only average delays. Therefore the maximum delay can not be evaluated, which is crucial for real time system dedicated to process control.

Many languages and tools are identified for exhaustive modelling, based on communicating timed automata, such as SDL [22], RdPT [10], Kronos [7], UPPAAL [11] and IF [8, 9]. IF is chosen because of the large opening of the models and the use of models to generate exhaustive execution graphs. Using IF, a system is described by a set of processes. A process is modelled by communicating timed automaton. The processes communicate themselves asynchronously by signals or shared data. Each process owns a FIFO queue (with or without loss) to manage incoming signals. From time execution flow point view, it is important to notice that IF considers only discrete time. At last, from an IF specification, it possible to generate a corresponding labelled transitions system (LTS). From the LTS and using specific tools, it is possible to evaluate all the temporal behaviour of the system.

3. Formal models for data acquisition system

3.1 System modelling

This section presents main principles of IF modelling for the data acquisition system:

- Element of the system is modelled by an IF process.
- Because communication is asynchronous, a synchronous reading operation is modelled by an emission of a signal *read_X()* and the waiting of a return signal *ret_X()* (c.f. figure 2).
- Only the temporal behaviour is modelled, the data values are not directly modelled and are used only for observation.

We now give modelling principles for each element. A physical sensor is modelled by an IF sensor process (c.f. figure2). A sensor process models the arrival law of the data coming from the physical sensor. Data are produced according to a production law fixed by the sensor itself. We consider periodic laws (with or without jitter), sporadic, or in gust [21] arrival laws.

A communication interface with physical sensor is modelled by an IF communication interface process (c.f. figure2). A communication interface process models the data storage policy (FIFO, LIFO) into buffers. A buffer is characterized by its size. When a buffer is full, we consider three cases: the data is lost or the first or the last memorized data is overwriting and so lost.

The process models the temporal behaviour of the task: a device driver is modelled by a single IF driver process. We assume that driver has a simple mono-task implementation. A driver process is characterized by the task activation mode (polling mode or event-driven mode) and by the data storage policy (FIFO or LIFO). In the polling mode, the driver process periodically checks the communication interface buffer. In event-driven mode, the driver sets a signal conveying a specific “sending condition” to the communication interface process. Then, after receiving a data, the communication interface process, checks the sending condition in order to send a new data to the driver.

A real time application is also modelled by an IF application process. An application process can work in two modes, either polling mode or event-driven mode (described in the previous part). We consider here a single task application.

Because IF transitions have a null duration, time flow is only considered through transition instants using clock guards. These guards may represent discrete intervals using specific IF constraint (*delayable*). Due to temporal IF semantic, modelling a processing duration (related to task execution time) requires addition of waiting states all over the various concerned processes. In fact execution time is viewed through responsive time. The responsive time is related to the tasks execution time which depends on execution CPU time (real execution time), the other concurrent tasks, the scheduling policy and the presence of possible delays related to shared resources access. Scheduling Analyses [1, 20] make possible to compute minimal and maximum bounds of these responsive times and then can be used in our proposed models.

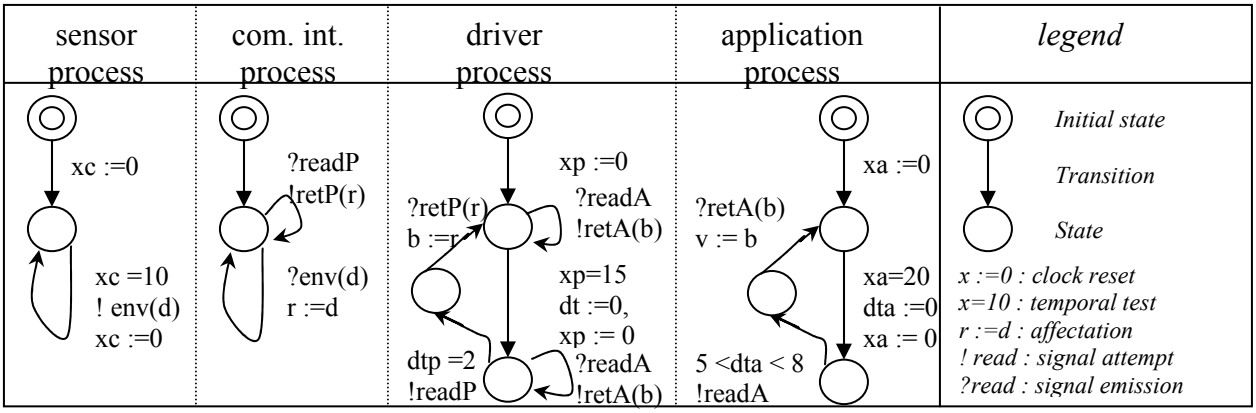


FIG. 2 – An IF model of a data acquisition system for a periodic emission (period Xc of 10), a polling driver process (Xp period of 15, lasted dtp of 2) and an application process in polling mode (Xa period of 20, lasted dta from 6 to 7).

3.1.1 QoS modelling

To evaluate QoS characteristics we propose two approaches. First approach is to introduce observation variables into the model. The second approach is based on non intrusive observers. The variables are counters instance number of signals (received by the application, sent by the sensor) and temporal counters (instants of production or consumption of data). An observer is a specific process that models a QoS characteristic. For instance, a delay is modelled via a specific process that behaviour is:

- Wait for signal "a new data is sent by the sensor".
- Wait for ticks.
- Wait for a signal "a new data is read by application".

The system is then instrumented with specific output signals ("a new data is sent by the sensor", "a new data is read by application").

The introduction of observation variables, such as a signal counter explodes the LTS generation. Indeed, this counter is incremented with each new data production (for instance a periodic low), following the time flow, it has infinite value. In order to perform an analysis, it is necessary to fix the system execution time to obtain a bounded LTS. The execution time must guarantee the generation of all cases. We proceed in two steps. We execute a first model without observation variables. The generated LTS must be finite (cyclic behaviour) and provides the execution time necessary to the model simulation with observation variable. It is necessary to notice that the introduced variables counters do not impact the system behaviour.

For the second approach to avoid infinite LTS, the observers have to be dynamically managed. An observer is created for a new event occurrence and is deleted when the QoS characteristic is computed (when the data is read by the application) or when the data is lost (full buffer). The second approach is easier to implement but limited to QoS related to single data (Maximum delay). The first one is more complex (two simulations) but more suitable for QoS related to data flow (maximum consecutive lost for a given time interval).

Due to the number of clocks, systems may generate LTS with a large size. For example, for simple architecture (mono task) and partly reduced (data not modelled) the generated LTS can go up to 1,000,000 states. The risks of combinatorial state explosion are real and will be illustrated thereafter.

4. Experimentations

4.1 Basic illustration

In order to validate and to illustrate the proposed approach, we present a first basic example :

- the physical sensor period *Tc* is set to 10 units of time
- the communication interface buffer size is 1
- the driver works in polling mode and has an a priori unknown period *Tp*, its buffer size is 1 with data overwrite if necessary, its responsive time is fixed as a constant 2 units of time
- the application period *Ta* is equal to 20 units of time and its responsive time considered as null
- At last, it is considered that all the activities of the system (sensor, driver, application) start at the same time.

The figure 3 represents the evolution of the maximum loss rate according to driver period. We consider the discrete interval of time [3, 42]. The shape of the obtained curve is relatively simple to interpret. The percentage of loss increases with the driver period. We notice three areas on the shape:

- in the first one from 3 to 18, the loss rate is equals to 50%. The data is produced (Tc=10) two times more than the application reads it (Ta=20).
- from the 18 to 38 the loss rate is higher (75%) because the driver is not executed as often as necessary (Td > Ta).
- new area, is for each application period (38,…). The values (18, 38, 58, ..) are a consequence of responsive time, equals to 2 units of time.

The figure 4 represents the evolution of the minimum and maximum delays according to the driver polling period. For the minimum delay, we notice three possible values equals to 0, 10 and 20 units of time. This curve can be obtained by analytical approaches. Using these techniques, either exact evaluation or bounds evaluation can be performed. For the first one, due to space restriction, we just illustrate how to evaluate when maximum delay is null. A null delay corresponds to a perfect synchronisation between production of the data by the sensor, the driver storage and the application consumption. These correspondences can be modelled by the following equation:

$$n1 * Tc = n2 * Tp + Cexec = n3 * Ta$$

- $n1 * Tc$ represents dates of production by the sensor.
- $n2 * Tp$ represents driver activation dates.
- *Cexec* represent the driver responsive time.
- $n3 * Ta$ represent application consumption dates.

To solve this equation, we have to search positive integer solutions for n1, n2 and n3. The solving of this kind of equations (known as diophantines) is relatively simple (based on the calculation of Greatest Common Factor). Thus it is possible, for each *Tp*, to find if this equation has an integer solution. If a solution exists the correspondence is possible and the delay is null.

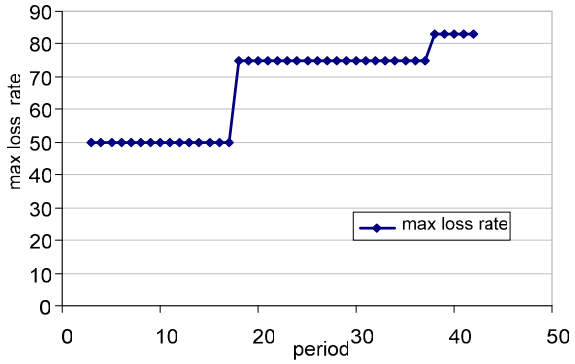


FIG. 3 – Maximum loss rate according driver period

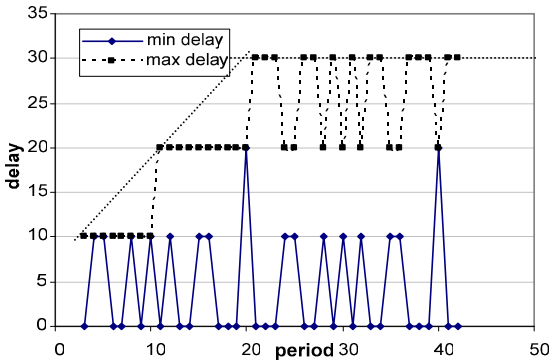


FIG. 4 – Minimum and maximum delay according driver period

To evaluate bounds (best and worst delay), it is possible to use RMA technique [19] :

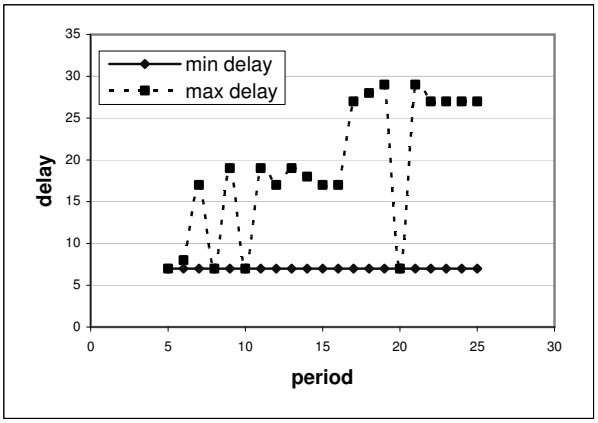
- minimum delay is equals to 0 (synchronisation of each elements).
- for maximum delay, they are two cases. If the driver is executed less often than the application (Tp < Ta), the maximum delay is equals to Tc+Tp. If the driver is executed more often than the application (Tp > Ta), either the data is lost or the data has a maximum delay equals to Tc+Ta.

This basic case shows the interest of an exhaustive evaluation (impact of responsive time on loss rate, common factors between the different periods) and realistic results (compare to RMA techniques). Moreover, exhaustive simulation tools bring the benefit of an automating (correctness, time) evaluation.

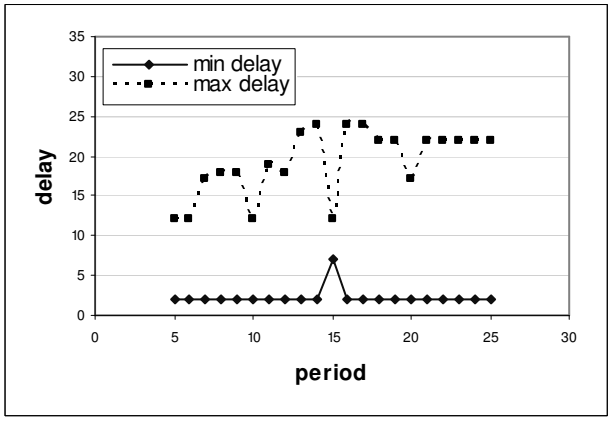
4.2 Realistic examples

In order to evaluate the interest of the proposed approach, we consider more complex systems. In the previous example, the application responsive time and its variability were neglected. Moreover, the previous case is very particular because of the common factor between sensor and application periods (10 and 20). We consider now the same case study but with different parameters. The results are presented on figures 5.x :

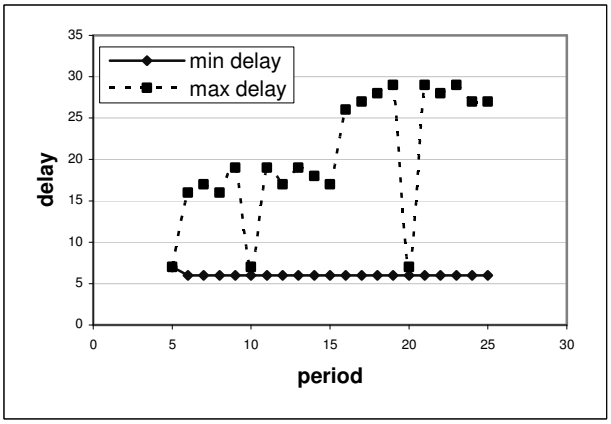
- 5a: hypothesis given by figure 4 and responsive time of the application is set to 7.
- 5c: hypothesis given by 5a and responsive time of the application is set to [6, 7].
- 5d: hypothesis given by 5c and responsive time of the driver is set to [1, 2].
- 5b: hypothesis given by figure 4 and application period is set to 15.



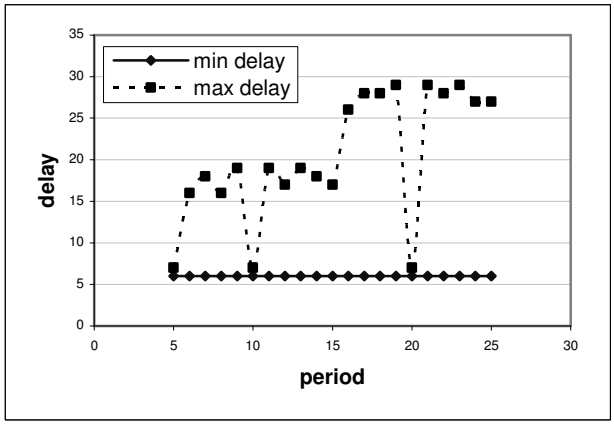
(a) Driver responsive time: 2
Application period: 20, Responsive time: 7



(b) Driver responsive time: 2
Application period: 15, Responsive time: 7



(c) Driver responsive time: 2
Application Period: 20, Responsive time in: [6, 7]



(d) Driver responsive time in: [1, 2]
Application Period: 20, Responsive time in: [6, 7]

FIG. 5 – Evolution of the delay according to the driver polling period for different requirements system

First of all, it is interesting to notice that the minimum and maximum delays of the figure 5d are included in those of the figure 5c, them even include in those of the figure 5a.The reason is the case 5d contains the case 5c that contains itself the case 5a.

Let us compare, figures 4 and 5a. Figure 4 is in fact a very particular case because it offers many points of synchronization (between the sensor, the driver and the application). This phenomenon explains the complex form of the results. The curves of the figure 5a thus are more smoothed. The minimum delay is now 7 (reading all the (20 k +7) units of times of a data produced by the sensor all 20k units of time). Some interesting points exist and correspond to particular synchronization between the application and the sensor.

The figure 5c is also smoothed. This illustration shows that the variability limits the impact of common factors. The figure 5d is very similar to the figure 5c, besides some small differences of +/- 1 unit on the minimum and maximum delays related to the influence of the interval [1, 2]. It must be noticed that there are for certain driver periods (7 and 17), the maximum delay is worst for [1, 2] than for 2.

Finally the figure 5b, shows an example of influence of the application parameters. The application period is changed from 20 units of time to 15 units of time: it involves considerable changes on the values obtained, but also of the curve shape.

4.3 Analysis

These experiments show the good expressivity of the models. In particular, they allow us to take into account the temporal properties defined as intervals. The analysis of these models remains possible and makes possible to obtain results that are not really intuitive and more realistic than result provided by RMA techniques.

A complete analysis of the considered cases is possible without considering communicating timed automata (the previous section). But, model based on communicating timed automata has the major interest to manage automatically the explosion of the number of cases to consider. Of course, the combinatorial state explosion is an important problem that must be considered. Table 1 presents the evolution of the states number for the generated LTS (in absence of observation variables). The examples of figure 5.a and 5.b do not raise difficulties. On the other hand, a more complex example, including a sporadic emission law, increases considerably the number of states on the generated LTS. For instance, changing the arrival laws of data to an interval [10,11] could not be explored because the analysis of the models (including observation elements) explodes (lightly) on the low power machine used to carry out the experiments.

example	production period : 10 driver period : 7 driver responsive time : 2 application period : 20 Application responsive time : 7	production period : 10 driver period : 7 driver responsive time : [1,2] application period : 20 Application responsive time : [6,7]	production period : [10,11] driver period : 7 driver responsive time : [1,2] application period : 20 Application responsive time : [6,7]
States number	387	1676	19331

TAB. 1 – States number evolution according to the example characteristics without observation elements

5. Conclusion

This paper proposes models of behavioural and temporal specification of a data acquisition system. These models make possible to characterize an acquisition driver from the QoS point of view. These models are expressed using IF timed automata and take into account the sensors, the communication interface, the driver and the application. The presented results demonstrate that the QoS properties evolution is complex and depends on several parameters, in particular of the driver architecture, but also of its execution environment.

This work can be used to qualify a software component for a particular context. It can also be useful to choose the driver parameters (polling period, buffers size).

The perspectives of this work are first to consider more complex architectures (multitasks, data multiplexing, layered drivers) and second to allow complex temporal models (continuous time, stochastic models).

At last, the problems of combinatorial state explosion raised in the previous paragraphs will be the subject of future works. We will evaluate the reductions using equivalence and/or the introduction of pre-computed analytical results.

References

[1] N. C. Audsley, A. Burns, M. F. Richardson, K. Tindell, and A. J. Wellings, "*Applying new scheduling theory to static priority pre-emptive scheduling*" Software Eng. J., vol. 8, no. 5, pp. 284–292, Sept. 1993.

[2] E. Robles, J. Held, *A comparison of Windows driver model latency performance on Windows NT and Windows 98* , Proc. Operating Systems Design and Implementation archive Proceedings of the third symposium on Operating systems design and implementation, 1999.

[3] M. Belarbi, J.-P. Babau, J.-J. Schwarz, "*Temporal Validation of Multitasking Real-Time Applications based on communicating Timed Automata*" „. *FDL'04*, Lille, 2004.

[4] M. Belarbi, J.-P. Babau, J.-J. Schwarz, "*Temporal Verification of Real-Time Multitasking Application Properties Based on Communicating Timed Automata*", Proc. in 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications, Budapest, 2004.

[5] A. Zahir, P. Palmieri, "*OSEK/VDX-operating systems for automotive applications*", in OSEK/VDX Open Systems in Automotive Networks, 1998

[6] S. Bornot , J.Sifakis. *An Algebraic Framework for Urgency,* Information and Computation 163,172-202 ,2000.

[7] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. *Kronos: "A model-checking tool for real-time systems"*. In Proc. of the 10th Conference on Computer-Aided Verification, Vancouver, Canada, 1998.

[8] Bozga, Dorel Marius. *Symbolic verification for communication les protocols.* Phd thesis, Verimag, University Joseph Fourrier, 1999.

[9] M. Bozga, S. Graf, L. Mounier, "*IF-2.0: A Validation Environment for Component-Based Real-Time Systems*", In Ed Brinksma, K.G. Larsen (Eds.) Proceedings of CAV'02 (Copenhagen, Denmark) LNCS vol. 2404 Springer-Verlag July 2002.

[10] G. Ciardo, and J. Muppala,; and Trivedi, K.S., "*SPNP: Stochastic Petri Net Package*", In International Conference on Petri Nets and Performance Models , 1989.

[11] A. David, G. Behrmann, K. G. Larsen, and W. Yi. "*A tool architecture for the next generation of uppaal*", Technical report, Uppsala University, 2003.

[12] Dedicated System, "*Comparison report between QNX NEUTRINO RTOSv6.2, VxWorks AE 1.1, Windows CE .NET 2.52 and Red Hat Embedded Linux Development Suite*", ELDS v1.1, Technical report, 2003.

[13] L. Waszniowski, and Z. Hanzálek, "*Analysis of Real Time Operating System Based Applications*", Proc. First International Workshop on Formal Modeling and Analysis of Timed Systems 2003, LNCS Springer, Marseille, 2003.

[14] F.J. Fokkink, N.Y. Ioustinova, E. Kesseler, J.C. van de Pol, Y.S. Usenko and Y.A. Yushtein, "*Refinement and verification applied to an in-flight data acquisition unit*", in Proc. 13th Conference on Concurrency Theory - CONCUR'02, Brno, Lecture Notes in Computer Science 2421, pp. 1-23, Springer, 2002.

[15] P.T. Hieu, S. Gérard and F. Terrier, "*Scheduling Validation for UML-modeled real-time systems*", *Proc. Workshop WiP (EuroMicro'03)*, 2003.
[16] *ISO/IEC Standard 9945-1: 1996 [IEEE/ANSI Std 1003.1, 1996 Edition] Information Technology—Portable Operating System Interface (POSIX)—Part 1: System Application: Program Interface (API) [C Language]*. IEEE ISBN: 1-55937-573-6, 1996.
[17] Time-Triggered Technology TTTech Computertechnik AG. "*Specification of the TTP/C protocol*". *,Technical report, Vienna*, Austria, July 1999.
[18] S. Vestal "*Modeling and Verification of Real-Time Software Using Extended Linear Hybrid Automata*" , Proc Fifth NASA Langley Formal Methods Workshop 2000.
[19] M. Klein & all "*A practionner's Handbook for Real-Time Analysis*", Kluwer Academic Publishers, 1993.
[20] C.L. Liu, J.W. Layland "*Scheduling algorithms for multiprogramming in a hard real time environment*" Association for Computing Machinary, 20 (1), 46-61,1973.
[21] Z. Mammeri., "*Expression and derivation of temporal constraints on real time applications*". In: "Journal Européen des Systèmes Automatisés", JESA-APII, Hermès, V. 32 N. 5-6, p. 609-644, 1998. (In French)
[22] Z. Mammeri. "*SDL - protocols and reactive systems Modelling* ", Hermès sciences, Julia 2000 (in French).
[23] J. Migge, A. Jean-Marie, N. Navet "*Timing analysis of compound scheduling policies : Application to posix1003.b* " Journal of Scheduling, Kluwer Academic Publishers, 6 (5), 457-482, 2003.
[24] P. Ramanathan,,. "*Overload Management in Real-Time Control Applications Using (m, k)-Firm Guarantee*". IEEE Transactions on Parallel and Distributed Systems 10(6): 549-559, 1999
[26] J-C Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, M. Sighireanu, "*CADP A Protocol Validation and Verification Toolbox*". 8th International Conference on Computer Aided Verification CAV, New Brunswick (USA), 1996.