

UML Profiles for Real-Time Systems and their Applications

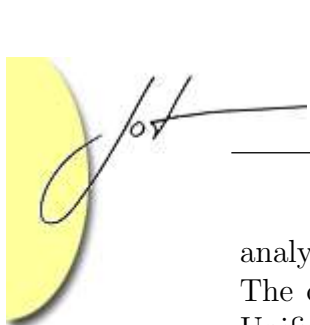
Abdelouahed Gherbi and Ferhat Khendek

Electrical and Computer Engineering Department
Concordia University, Montreal, Canada

Real-time systems (RTS) have strict timing constraints and limited resources. The satisfaction of RTS timing constraints is required for their correction. In order to reduce the cost due to late discovery of design flaws and/or violations of timing constraints of RTS as well as to speed up their development to cope with time-to-market requirements, it is important to validate, at early stages of the development process, the functional and nonfunctional properties of RTS. In addition, RTS complexity is continuously increasing which makes their design very challenging. UML, a graphical object-oriented modeling language, is suitable to deal with this complexity. UML also supports predictive, quantitative analysis through its real-time profiles. The objective of this paper is to review the most important UML profiles for real-time from the academia, the industry and/or standard organizations; and the research activity that revolves around these profiles.

1 INTRODUCTION

Nowadays real-time systems are used in a wide variety of applications, including avionics, automotive, patient monitoring, to mention just a few. A Real-time System (RTS) is constrained to carry out its functionality under (strict) timing constraints. With respect to the latter, RTS are often classified into hard or soft real-time systems. In the former class, a missed deadline is synonymous to disastrous consequences such as loss of life or property. Soft RTS, such as a video play back system, tolerate some occasional deadline misses. RTS are generally embedded within a nondeterministic and highly concurrent environment, for instance, a cruise control system within a car. This environment generates streams of asynchronous and concurrent events having different characteristics, i.e., periodic, sporadic etc. RTS should react to these events timely. Consequently, RTS should be concurrent to maximize their responsiveness as well as to use efficiently their computing resources. Moreover, a RTS is generally a distributed system that takes advantage of real-time networking facilities such as CAN buses or FDDI. Finally, RTS are often used to monitor and/or control very critical systems which require high levels of availability and dependability. These characteristics make RTS very complex and their design a daunting challenge. Therefore, an adequate modeling language for RTS should enable to manage their inherent complexity and support quantitative



analysis in order to validate their nonfunctional properties.

The object paradigm is very effective to cope with the software complexity. The Unified Modeling Language (UML), which is the *de facto* standard object-oriented modeling language for general-purpose software, is gaining interest among the real-time community. It is therefore interesting to consider how well UML is adapted to the real-time context. One important feature of UML stems from its built-in extensibility mechanisms: stereotypes, tag values and profiles. These allow to adapt UML to fit the specificities of particular domains or to support a specific analysis. Therefore, numerous UML profiles have been proposed in the academia, the industry and/or by standard organizations in order to accommodate the features of real-time software.

In this paper, we address the capabilities of UML in expressing the real-time requirements through some profiles specifically designed for this context. In particular, we review the main existing real-time UML profiles and the research that revolves around them. It is not the objective of the paper to give a comprehensive and thorough description of the profiles involved but we limit ourselves to the main concepts and show how these profiles are related to the real-time context.

The remaining part of this paper is structured as follows: In Section 2, we focus on some of the new built-in features of UML 2.0 that fit the requirements of real-time systems. Section 3 is devoted to the OMG UML profile for Schedulability, Performance and Time [22] as well as the research that revolves around it in order to support quantitative analysis at the design stage. In Section 4, we review an emerging OMG profile that enables the designer to define new quality of service requirements and their corresponding analysis. The Rational UML profile for real-time systems, UML-RT, is considered in Section 5 whereas TURTLE profile is considered in Section 6. Another industrial profile aiming at combining UML and SDL is presented in Section 7. Last but not the least, we reserve Section 8 to present succinctly other UML profiles from the literature targeting real-time systems. We present a comparison of the main important UML profiles for real-time and discuss some research issues in Section 9. Finally, we conclude in Section 10.

2 REAL-TIME FEATURES IN UML 2.0

UML is a graphical, object-oriented modeling language which allows to specify and document the artifacts of software systems. UML is widely used to express the general-purpose software design models. Real-time software presents, moreover, some specific characteristics. In addition to the timing constraints, an important characteristic of real-time software stems from the interaction with the environment. This interaction is inherently nondeterministic and highly concurrent. UML 2.0 presents some features that support real-time aspects [5], [10]. For example, it supports the modeling of concurrency by providing some concepts, including active objects, concurrent composite states and concurrent operations. In order to express timing constraints, UML 2.0 provides two data types: `Time` and `TimeExpression`.

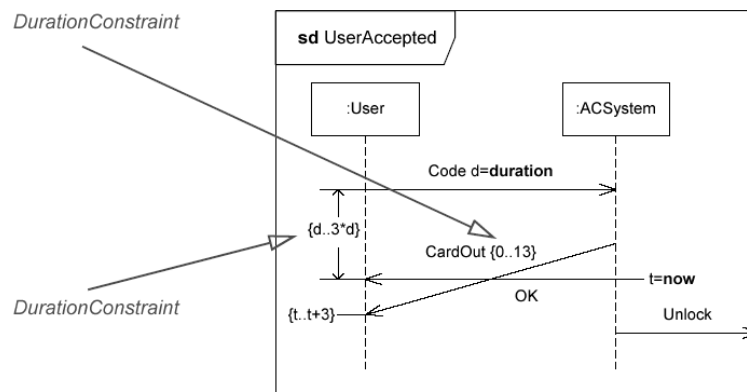


Figure 1: Sequence Diagram with Timing Constraints [23]

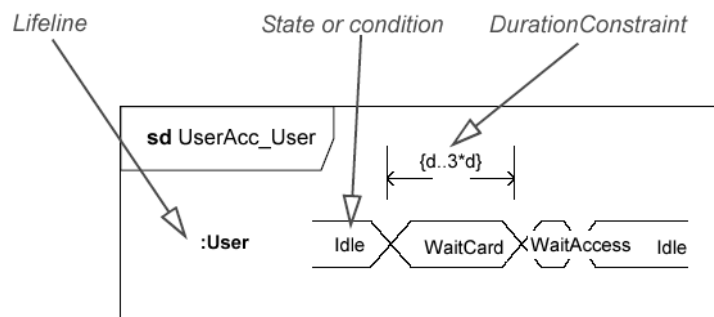


Figure 2: Timing Diagram Example [23]

These timing statements can be used either in state diagrams or sequence diagrams as depicted in Figure 1. Moreover, UML 2.0 introduces a new diagram called Timing Diagram to allow reasoning about time and visualize conditions or state changes over time. Figure 2 illustrates an example of a Timing Diagram. UML is, however, better adapted to the real-time software context through its built-in extensibility mechanisms, particularly its profiles. The most important ones as well as the research activity that these profiles drive are the focus of the following sections.

3 UML PROFILE FOR SCHEDULABILITY, PERFORMANCE AND TIME

The UML profile for real-time modeling, formally called the UML profile for Schedulability, Performance and Time (UML/SPT), was adopted by the OMG in 2002 [22]. This increased the interest in the use of the object-oriented technology and UML in particular to model and build real-time systems [26]. UML/SPT is a framework to model quality of service, resource, time and concurrency concepts and to support predictive quantitative analysis of UML models. Actually, it provides the user (modeler) with a set of stereotypes and tagged values in order to annotate the UML models. Quantitative analysis (schedulability and performance analysis) can then

be applied to these (predictive) UML models.

The structure of the UML/SPT, as illustrated in the Figure 3, is composed of a number of sub-profiles:

- The General Resource Model (GRM) package is the core of UML/SPT. It is further partitioned into three packages:
 - RTresourceModeling for the basic concepts of quality of service and resource.
 - RTConcurrencyModeling for concurrency modeling.
 - RTtimeModeling for time and time-related mechanisms modeling.
- The Analysis Modeling package defines the analysis sub-profiles, including:
 - PProfile for performance analysis.
 - SProfile for schedulability analysis.

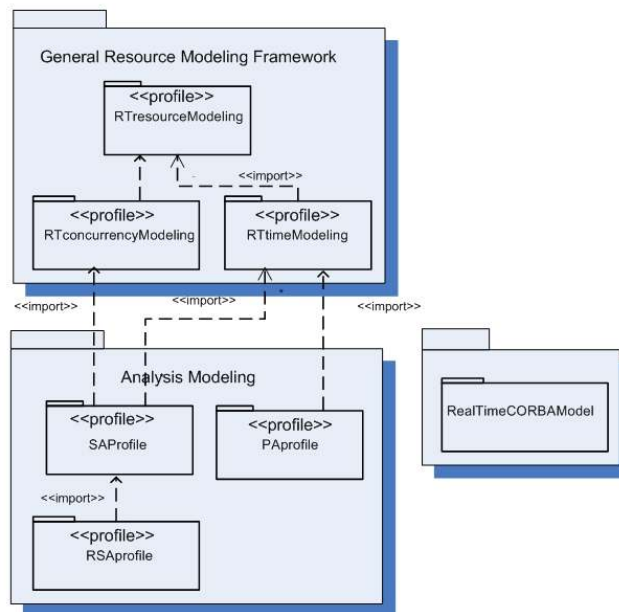


Figure 3: The Structure of UML/SPT Profile

The standard UML/SPT is too large to be thoroughly covered in this paper. Therefore, we just illustrate the features that are directly related to the real-time modeling. The interested reader is referred to [22] for an exhaustive description of the standard. With regard to time modeling, the sub-profile [RTtimeModeling](#) defines a metamodel representing time, as depicted in Figure 4, and time-related mechanisms, as illustrated in Figure 5. The profile provides a set of stereotypes and associated tagged values that the modeler could apply to UML modeling elements to specify time values [«RTtime»](#), time-related mechanisms such as [«RTclock»](#), [«RTtimer»](#),

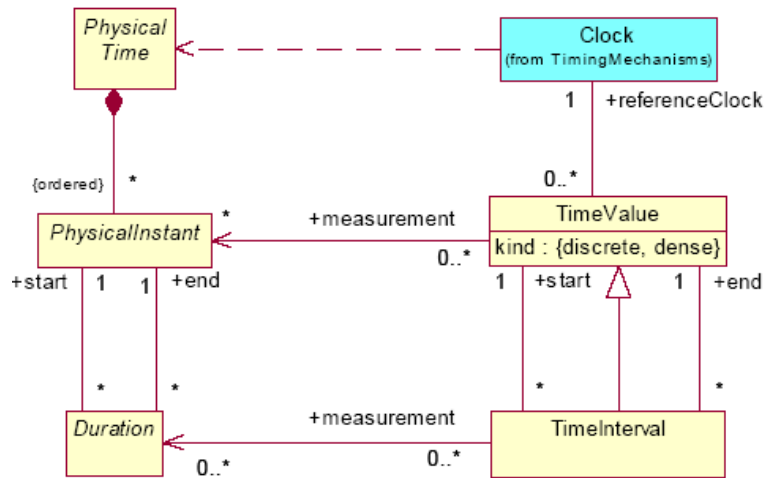
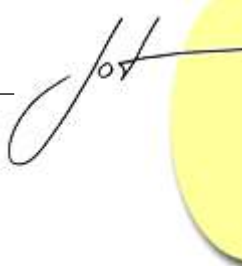


Figure 4: Time Modeling in UML/SPT [22]

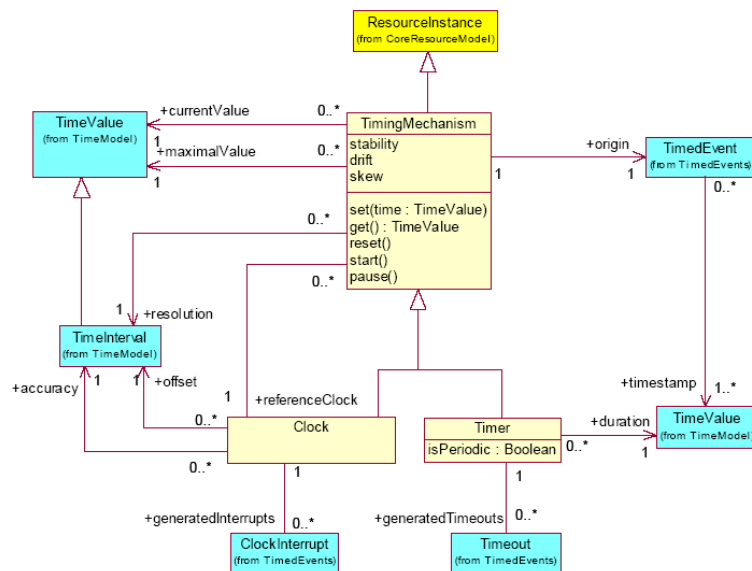


Figure 5: Timing Mechanisms In UML/SPT [22]

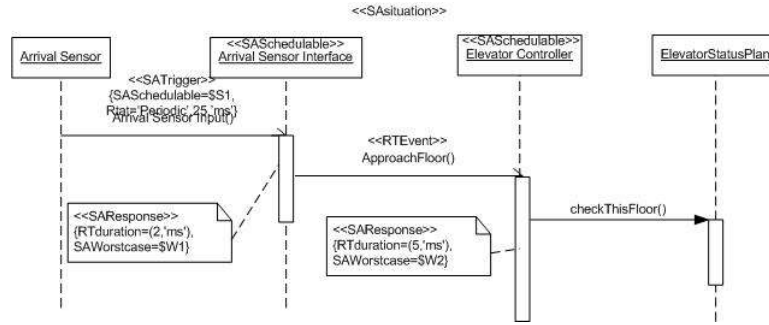


Figure 6: Sequence Diagram Annotated with UML/SPT Stereotypes[22]

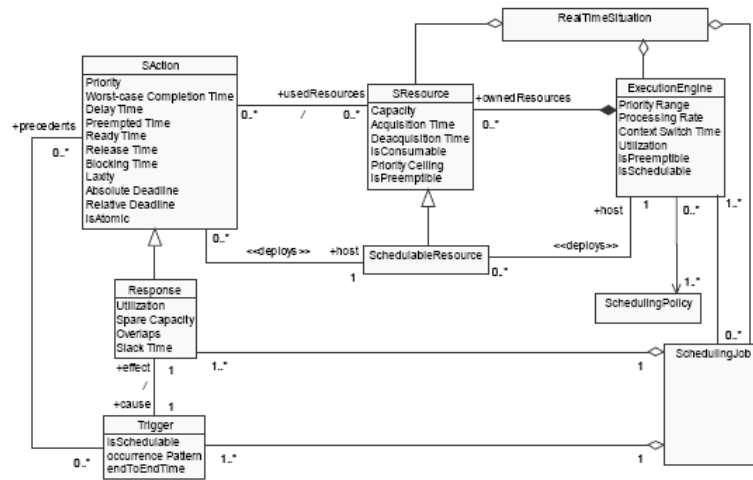


Figure 7: UML/SPT Schedulability Analysis Metamodel [22]

`<<RTtimeout>>` or timing constraints `<<RTdelay>>`, `<<RTintervale>>`. For example, Figure 6 illustrates a UML sequence diagram modeling the behavior of an elevator control system in reaction to the arrival sensor event. This sequence diagram is annotated with UML/SPT stereotypes to model the periodicity of the event, the timing constraints on the actions of the different components of the system etc.

The `Saprofile` sub-profile is designed to support schedulability analysis of UML models. Figure 7 depicts the metamodel defined in UML/SPT for the main concepts involved in the schedulability analysis: the execution engine, threads (task or process), shared resources, external events and the response of the system to the external events. Correspondingly to these concepts, a set of stereotypes and their associated tagged values is defined in UML/SPT. A sample of these is presented in the Table 1. The application of these stereotypes on a collaboration diagram is illustrated in Figure 8.

The ability to undertake quantitative analysis at early phases of the software development process is important to reduce the cost. Consequently, numerous research activities are dedicated to the integration of software design modeling and performance analysis [3]. In particular, UML/SPT is at the heart of many active

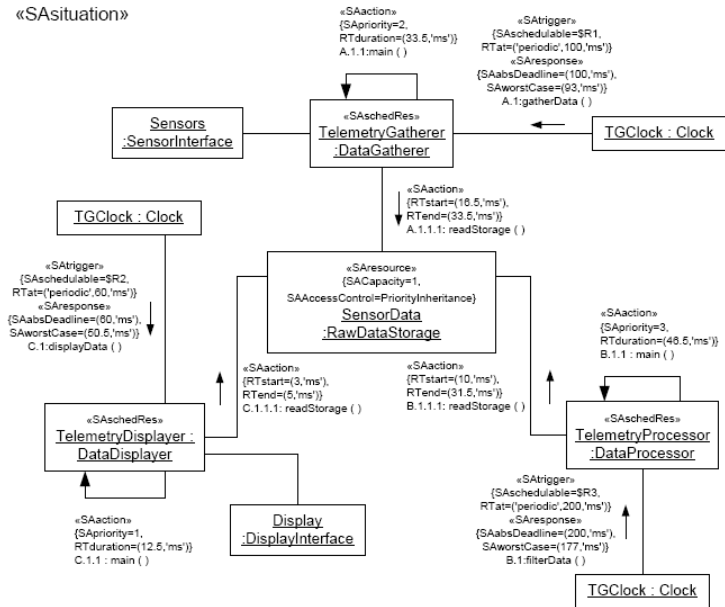
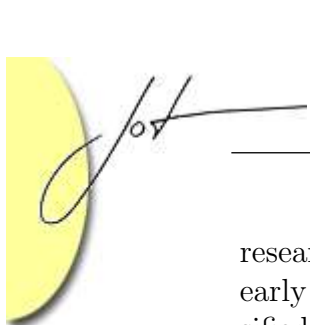


Figure 8: Collaboration Diagram Annotated for Schedulability Analysis [22]

Steriotype	Real-time Concept	UML Model Element
<<SASituation>>	Real-time situation	Collaboration, Sequence diagrams
<<SATrigger>>	Event	Message, Stimulus
<<SAResponse>>	Response	Method, Action
<<SAAction>>	Action	Method, Stimulus, Action
<<SASchedulable>>	Task, Thread	Instance, Object, Node
<<SAResource>>	Resource	Instance, Class, Node
<<SAEngine>>	CPU, Processor	Object, Class, Node

Table 1: UML/SPT Common Stereotypes for Schedulability Analysis



research initiatives aiming at the integration of predictive quantitative analysis to early stages of software development process. These research works could be classified into two main tracks: the performance track aiming at deriving performance models, including queueing networks [4], layered queue networks or extended queue networks [34], or other formalisms such as general stochastic petri nets. These models enable achieving performance analysis of UML models. The schedulability track, on the other hand, includes many research initiatives that aim at integrating the schedulability theory with object-oriented real-time design [16], [17], [27], and [28]. Moreover, other research activities focus on the fundamentals of modeling concepts in UML/SPT and address its capabilities and limitations [7].

4 UML PROFILE FOR QUALITY OF SERVICE

The UML profile for modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (UML/QoS) was adopted by the OMG in 2004 [24]. It is an emerging UML profile which aims at capturing the concept of quality of service at large. It allows the definition of an open variety of quality of service requirements and properties [8].

We consider that UML/QoS is relevant for real-time software modeling. This is because it is introduced to complement the aforementioned UML/SPT profile. However, while UML/SPT is tailored to fit performance and schedulability analysis, UML/QoS allows the designer to define any set of quality of service requirements and carry out any specific analysis that could be relevant for the safety-critical aspect of real-time software. This was demonstrated in [6], where a quality model has been defined to drive a dependability and performability analysis of an embedded automation system.

In a nutshell, UML/QoS could also be used to annotate UML diagrams. In contrast to UML/SPT, UML/QoS proposes a procedure that consists in three steps:

- Definition of QoS characteristics: The new user-defined QoS characteristics could leverage, through specialization, the general QoS characteristics catalogue defined in UML/QoS. This catalogue is organized, as depicted in Figure 10, in categories **Performance**, **Dependability**, **Security**, **Integrity**, **Coherence**, **Throughput**, **Latency**, **Efficiency**, **Demand**, **Reliability** and **Availability**. In particular, Figure 9 illustrates the category that includes the Latency QoS characteristics. The latter might be leveraged to adequate the real time context. Notice that the QoS characteristics are templates classes having parameters. The later have to be instantiated in the next step.
- Definition of the quality model: The QoS characteristics parameters should be assigned actual values. This is done through the definition of quality characteristics bound class and template bindings. The UML model containing the binding information and the bound classes is called the Quality Model. An example of a Quality Model is illustrated in Figure 11.

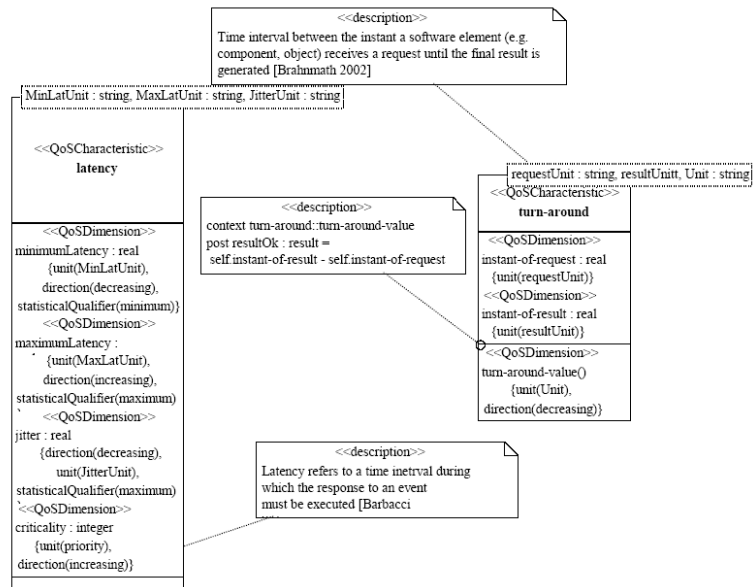


Figure 9: Latency QoS Characteristics Category [24]

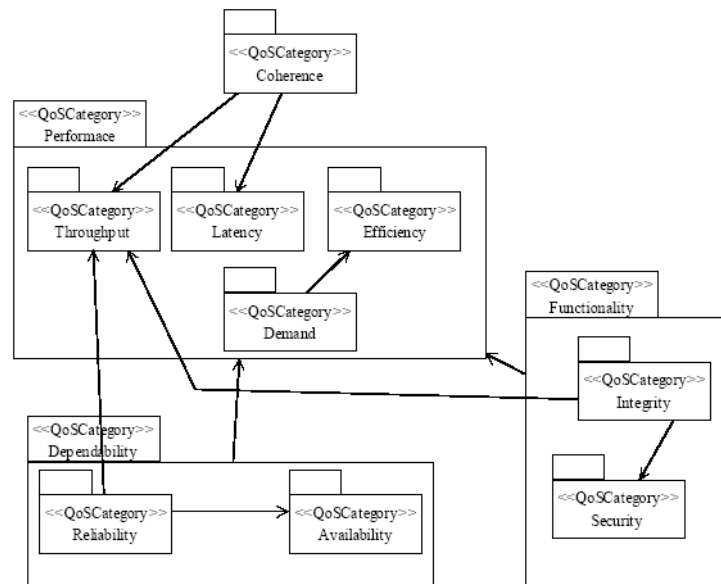


Figure 10: Qos Characteristics Categories Catalogue [24]

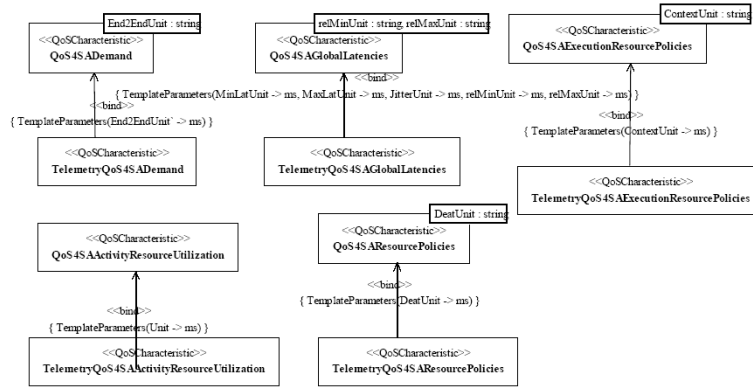


Figure 11: An Example of Quality Model [24]

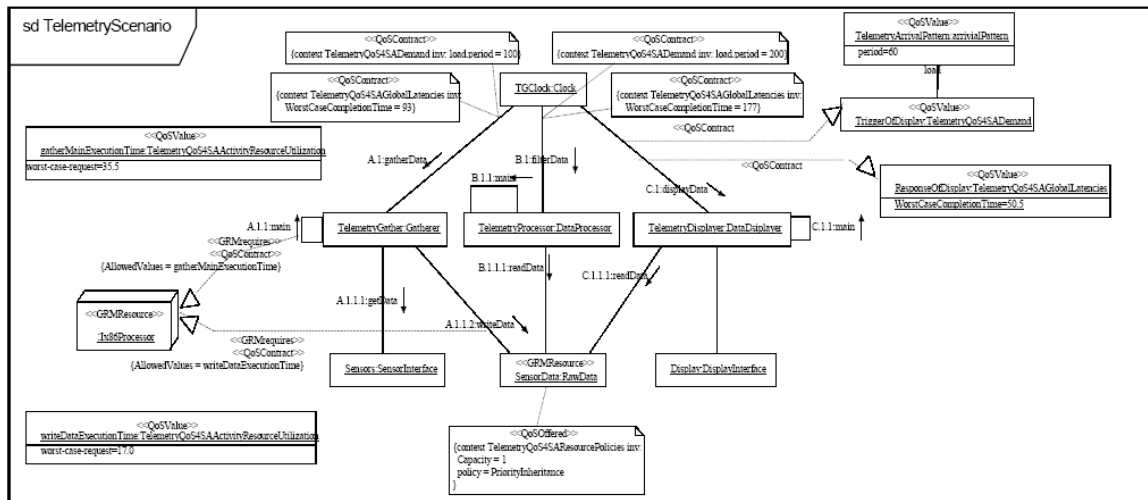
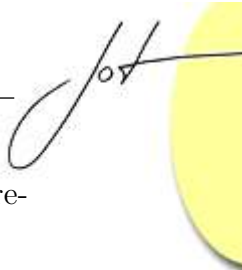


Figure 12: UML Collaboration Diagrams Annotated with QoS [24]



- The last step is the UML models annotation using quality of service requirements. This is illustrated by the Figure 12.

We believe that UML/QoS is not supported yet by a CASE tool. We are expecting, however, that it will be investigated more in the near future because of its flexibility in terms of analysis with respect to UML/SPT.

5 UML-RT PROFILE

UML-RT is a real-time profile developed by Rational Software [30]. It uses the UML built-in extensibility mechanisms to capture the concepts of defined in the Real-time Object Oriented Modeling (ROOM) Language [29]. In contrast with the two previous profiles, UML/SPT and UML/QoS, UML-RT is not just meant to annotate a design model with information allowing for quantitative analysis. It is a modeling language of its own. Indeed, UML-RT allows the designer to produce models of complex, event-driven and possibly distributed real-time systems. However, UML-RT does not support time and timing constraints modeling. UML-RT is supported by a CASE tool called RationalRT that allows for automatic code generation by compiling the models and linking them with a run-time system.

UML-RT includes constructs to model the structure and the behavior of real-time systems:

- **Structure Modeling:** UML-RT provides the designer with entities called capsules, which are communicating active objects. The capsules interact by sending and receiving messages through interfaces called ports. Furthermore, a capsule may have an internal structure composed of other communicating capsules and so on. This hierarchical decomposition allows the modeling of complex systems. Figure 13 illustrates the structure model of an elevator control system using UML-RT.
- **Behavior Modeling:** The behavior is modeled by an extended finite state machine, and it is visualized using UML state diagrams. These state machines are hierarchical since a state could be decomposed into other finite state machines. A message reception triggers a transition in the state machine. Actions may be associated with transitions or the entry and/or the exit of a state. For example, Figure 14 illustrates a model for the behavior of the elevator controller component in the Elevator Control System depicted in Figure 13.

Similarly to the two previous UML profiles, UML-RT lacks formal foundations. UML-RT is, however, a basis for a very active research work on schedulability analysis applied to real-time software design models. Indeed, while the CASE tool RationalRT allows an automatic code generation, it does not take into account timing constraints. Therefore, the research reported in [28] was a first attempt to integrate the real-time schedulability theory with object-oriented design targeting real-time

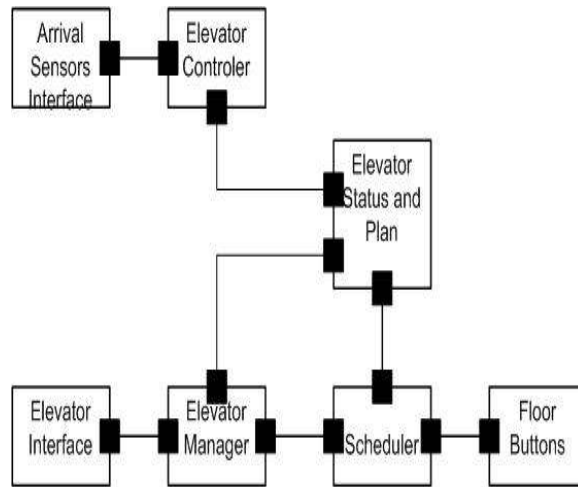


Figure 13: Elevator Control System UML-RT Model

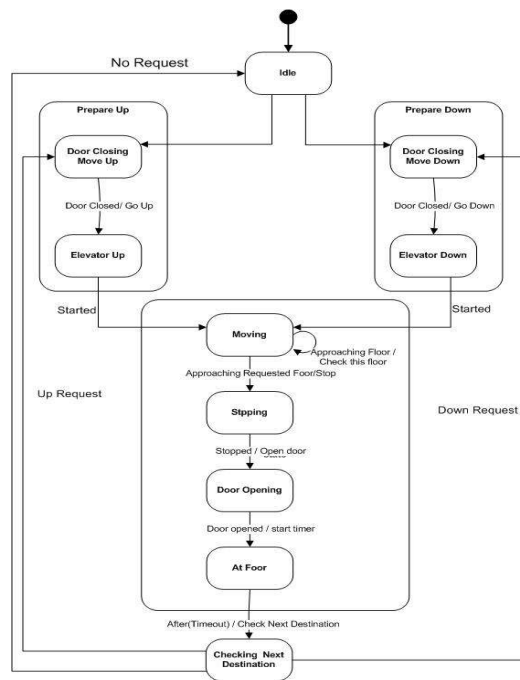


Figure 14: Finite State Machine of an Elevator Controller




Tclass Id 	<i>Tclass identifier</i>
Attributes	All attributes, except gates
Gates	Gates may be declared as public (+) , private (-) , or protected (#)
Methods	Methods, including a constructor
Behavioral Description	Activity diagrams may use inherited and locally defined attributes, gates and methods

Figure 15: A TURTLE TClass Structure [1]

systems. This work showed how the fixed-priority scheduling theory could be applied to design models using UML-RT. An other approach for a schedulability-aware mapping of UML-RT models to scenario-based multi-threaded implementation was reported in [16], [17], and [19]. Finally, a slightly different approach to apply real-time scheduling techniques to obtain real-time implementations from UML-RT models is given in [13], [14], [18], and [20].

6 TURTLE PROFILE

The Timed UML and RT-LOTOS Environment, TURTLE, is a UML profile aiming at the formal validation of complex real-time systems [1]. TURTLE uses UML's extensibility mechanisms to enhance UML structuring and behavioral expression power. In addition, TURTLE has a strong formal foundations. Actually, its formal semantics is expressed by means of a mapping to RT-LOTOS. This enables a formal validation as well as a simulation of the UML models.

Similarly to UML-RT, TURTLE does not annotate UML models using stereotypes but provides concepts and operators to express the model itself. TURTLE essentially allows the description of the structure/architecture as well as the behavior of the system using an extension of the UML class/object and activity diagrams. The main extensions brought by TURTLE are the following:

- **Structural Extensions:** TURTLE introduces the concept of TClass which has special attributes called Gates (see Figure 15). These are used by TClass instances, TInstances, to communicate and are specialized into InGate and OutGate. In addition, TURTLE introduces stereotypes called composition operators (see Figure 16). These are used to explicitly express parallelism, synchronization, and sequence relationships between TClasses.
- **Behavioral Extensions:** The behavior of a TClass is expressed using activity diagrams extended with logical and temporal operators (see Figure 17). These operators allow expressing synchronization on gates with data exchange. Moreover, TURTLE enables expressing temporal non-determinism and different sorts of delays (deterministic, nondeterministic).

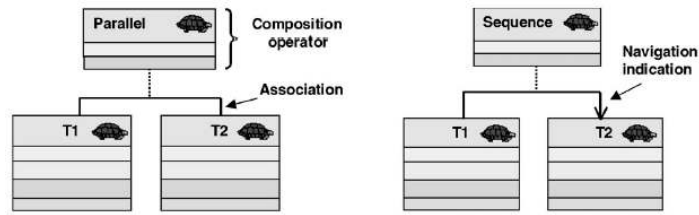


Figure 16: Some TURTLE's Composition Operators [1]



Figure 17: TURTLE's Temporal Operators[1]

TURTLE is supported by a toolkit composed of TTool [33] and RTL [25]. These are used by the designer to build a design, to run the simulation and to perform a reachability analysis for the validation of the system.

Finally, TURTLE has been recently extended to fit the requirements of distributed and critical system. The objective is to enable the definition of components and their deployment; and to study their properties at early stages of the software development process. This is done using a formal definition of the deployment diagrams, which are the most suitable for distributed architecture description. Therefore, TURTLE has been extended to take into account deployment diagrams. The obtained profile is called TURTLE-P [2], which addresses the concrete description of communication architectures. TURTLE-P allows the formal validation of the components and deployment diagrams through its foundations in RT-LOTOS.

7 SDL COMBINED WITH UML

The ITU-T recommendation Z.109, SDL Combined With UML [15], is actually a UML profile for SDL. It defines a specialization of a subset of UML and a one-to-one mapping to a subset of SDL. Thus, Z.109 has SDL as a formal semantics. This is intended to provide the designer with a combination of UML and SDL. We give in the sequel just an outline of the main concepts in this recommendation. For a comprehensive description of this profile, the reader is referred to [15]. In addition, a more readable presentation of this recommendation along with an illustrative example, a specification of an Automatic Teller Machine (ATM) using Z.109, can be found in [21].

Basically, Z.109 gives a UML model for the main concepts of SDL and provides with the corresponding stereotypes. In the following, we highlight the main concepts of SDL in Z.109 having a corresponding stereotype:

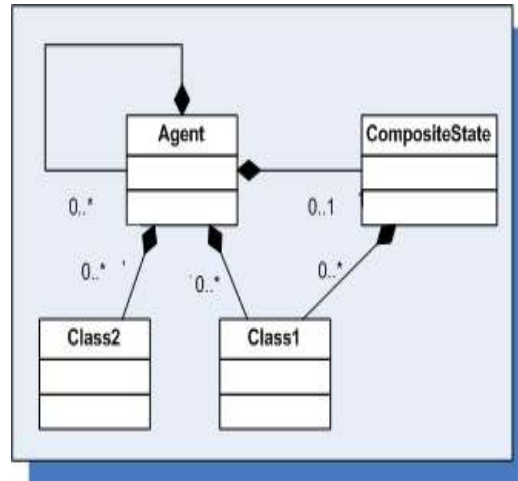


Figure 18: UML Model of the basic SDL Concepts

- **Agent** An SDL system is composed of agents connected through channels. An agent has a state machine and a internal structure composed hierarchically of other agents. Moreover, an agent can be (its kinds) a process, a bloc or a system. Z.109 provides the UML model representing these concepts as illustrated in Figure 18. In particular, an agent type is mapped into a class of active objects and its kind is stereotyped `<<system>>`, `<<block>>` or `<<process>>`.
- **Gates and Interface:** The agents communicate through gates by sending signals or requesting a procedure, which together, the signals and procedures, compose its interface. The latter is mapped into a UML interface and the former are stereotyped `<<signal>>` and `<<procedure>>`.
- **State Machine:** An SDL agent state machine is mapped to an UML state machine.
- **Package:** UML packages are used to represent SDL packages.

Finally, this profile has been implemented in the Telelogic CASE tool Telelogic TAU 3.5 [32].

8 OTHER PROFILES

In addition to the aforementioned UML profiles, there are several other proposals from the academia. These include mainly the following:

- A profile compliant to UML/SPT is presented in [12] as part of the OMEGA project [31]. This is a framework for UML-based real-time modelling allowing for the analysis and verification of time and scheduling aspects. In particular, it consists of a set of timed-events primitives. The formal semantics of these primitives is expressed in terms of timed automata with urgency.

- An UML profile based on an extension of OCL 2.0 metamodel for the specification of real-time constraints in OCL is presented in [9]. The formal semantics of this profile is given by means of a mapping to time-annotated temporal logic formulae expressed in CTL, which allows the formal verification of properties.

9 DISCUSSION

In this section, we compare between the aforementioned profiles according to criteria, including formal foundation, expressiveness and tool support. We discuss also some of the research issues related to UML/SPT.

From the previous review, we can distinguish, with respect to the formal foundation criterion, two groups of profiles: the first group includes UML/SPT, UML/QoS and UML-RT. These profiles lack rigorous, formal foundations. In particular for UML/SPT, this important limitation was pointed out in [11], for instance. The second group of profiles presents some formal semantics such as RT-LOTOS for TURTLE and SDL for Z.109.

As for the profiles expressiveness, we notice that OMG's profiles, namely UML/SPT and UML/QoS, introduce models for the concepts of time, resource and quality of service characteristics respectively using stereotypes to apply annotations to UML design models while the others; UML-RT, TURTLE and Z.109; do not express explicitly timing constraints but introduce new modeling elements and constructs such as capsules in UML-RT, TClass in TURTLE to build the model itself.

The ultimate objective of UML profiles for real-time is to embed UML in an integrated framework allowing the design, analysis and synthesis of real-time software. This aims at reducing the cost and coping with time-to-market requirements. While there already exist some tool support for the majority of the aforementioned profiles, a full integrated framework is however still lacking. This is particularly true for the OMG standard UML/SPT.

In our point of view there is a methodological problem that hinder the integration of UML/SPT in an integrated framework for the design, the validation and the synthesis of real-time software. Indeed, it is not specified in the standard specification how to use the profile. The UML "multi-view" modeling approach where different diagrams are used to capture different perspectives of the system, and the cross-cutting aspect of UML/SPT annotations through the UML model may lead to a serious consistency problem. Indeed, the consistency of UML/SPT annotations throughout the different views is not guaranteed. In addition, an effective and correct mapping of the UML/SPT annotations to a suitable model for schedulability analysis is still lacking. We are interested in these issues in our current research. Table 2 summarizes the comparison of the different UML profiles for real-time systems.

	Issuer	Tool Support	System	Analysis	Expressiveness	Formal Samantics
UML/SPT	Industry (OMG)	Rhapsody (iLogix) RT Studio (Artisan)	generic and real-time systems	Performance Schedulability	Time, Resource Concurrency	No
UML/QOS	Industry (OMG)	?	generic and real-time systems	User-defined	QoS	No
UML-RT	Industry IBM/Rational	RationalRT	event-driven real-time systems	Schedulability	Characteristic	No
TURTLE	Academia	TTool RTL	Distributed Critical Systems	No	Capsules, ports Async messages	No
Z.109	Industry ITU-T	Telelogic Tau 3.5	Telecommunication Critical Systems	No	Synch, Parallele delays operators	RT-LOTOS
OMEGA-RT	Academia	OMEGA tool set	Real-time Systems	Timing Scheduling	SDL concepts	SDL
OCL Profile	Academia	?	generic RT Systems	Verification	Timed events	Timed automata with urgency
					Real-time constraints	CTL

Table 2: UML Profiles for Real-Time Systems

10 CONCLUSION

The increasing complexity of nowadays ubiquitous real-time systems requires using an adequate modeling language. UML, which is a widely used graphical object-oriented modeling language, has proved to be effective and suitable for general-purpose software. Is it suitable for the real-time context where the system nonfunctional properties, including the timing behavior, are as important as its functionality?

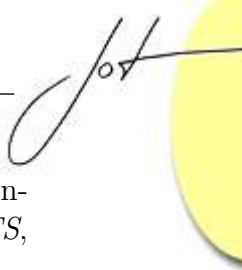
In this paper, we have reviewed the means that UML provides to cope with the real-time software requirements: the UML real-time profiles. We have paid special attention to the very active research activity that revolves around UML and its real-time profiles, in particular the OMG's standard UML/SPT. The latter is at the heart of many research initiatives to integrate quantitative analysis, performance and schedulability analysis, at early stages of the software development process.

While the model-based performance analysis has achieved some interesting results, we observe that the schedulability analysis and its integration into a design framework based on UML/SPT is still under-investigated. We are currently undertaking a research work to establish the basis of an integrated framework for a UML/SPT-based design modeling, schedulability analysis and implementation synthesis targeting embedded and real-time software systems.

Acknowledgments: This work has been partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

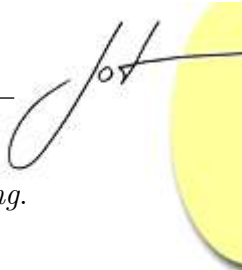
- [1] Ludovic Apvrille, Jean-Pierre Courtiat, Christophe Lohr, and Pierre de Saqui-Sannes. TURTLE: A Real-Time UML Profile Supported by Formal Validation Toolkit. *IEEE Trans. Software Eng.*, 30(7):473–487, 2004.
- [2] Ludovic Apvrille, Pierre de Saqui-Sannes, and Ferhat Khendek. TURTLE-P: A UML Profile for the Formal Validation of Critical and Distributed Systems. To appear in *Software and Systems Modeling*, Springer 2005.
- [3] Simonetta Balsamo, Antiniscia Di Marco, Paola Inverardi, and Marta Simeoni. Model-Based Performance Prediction in Software Development: A Survey. *IEEE Trans. Software Eng.*, 30(5):295–310, 2004.
- [4] Simonetta Balsamo and Moreno Marzolla. Performance Evaluation of UML Software Architectures with Multiclass Queueing Network Models. In *Fifth International Workshop on Software and Performance*, Palma, Illes Balears, Spain, July 2005.
- [5] Kirsten Berkenkotter. Using UML 2.0 in Real-Time Development: A Critical Review. In *SVERTS*, pages 41–54, San Francisco, USA, October 2003.



- [6] Simona Bernardi and Dorina C. Petriu. Comparing two UML Profiles for Non-functional Requirement Annotations: the SPT and QoS Profiles. In *SVERTS*, Lisbon, Portugal, October 2004.
- [7] C.M. Woodside and D.C. Petriu. Capabilities of the UML Profile for Schedulability Performance and Time (SPT). In *Workshop SIVOES-SPT held in conjunction with the 10th IEEE RTAS'2004*, Toronto, Canada, May 2004.
- [8] Miguel A. de Miguel. General Framework for the Description of QoS in UML. In IEEE Computer Society, editor, *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)*, pages 61–70, Hakodate, Hokkaido, Japan, May 2003.
- [9] Stephan Flake and Wolfgang Mueller. A UML profile for real-time constraints with the OCL. In Jean-Marc Jézéquel, Heinrich Hussmann, and Stephen Cook, editors, *UML 2002 - The Unified Modeling Language. Model Engineering, Languages, Concepts, and Tools. 5th International Conference, Dresden, Germany, September/October 2002, Proceedings*, volume 2460 of *LNCS*, pages 179–195. Springer, 2002.
- [10] S. Gérard and F. Terrier. *UML for Real: Which Native Concepts to Use*, chapter UML for Real-Time, pages 17–51. Kluwer Academic Publishers, 2003.
- [11] Susanne Graf and Ileana Ober. How useful is the UML profile SPT without semantics. In *SVERTS*, Lisbon, Portugal, October 2004.
- [12] Susanne Graf, Ileana Ober, and Iulian Ober. Timed annotations with UML. In *SVERTS*, San Francisco, USA, October 2003.
- [13] Z. Gu and K. G. Shin. Synthesis of Real-Time Implementation from UML-RT Models. In *2nd RTAS Workshop on Model-Driven Embedded Systems (MoDES '04)*, Toronto, Ontario, Canada, May 2004.
- [14] Zonghua Gu and Zhimin He. Real-Time Scheduling Techniques for Implementation Synthesis from Component-Based Software Models. In *Accepted to ACM SIGSOFT International Symposium on Component-Based Software Engineering (CBSE 2005)*, St. Louis, MO, 2005.
- [15] ITU-T. SDL combined with UML. 2000. ITU-T recommendation Z.109.
- [16] Saehwa Kim, Sukjae Cho, and Seongsoo Hong. Automatic Implementation of Real-Time Object-Oriented Models and Schedulability Issues. In *Workshop on Object-Oriented Real-Time Dependable Systems (WORDS)*, pages 137–141, Rome, Italy.
- [17] Saehwa Kim, Sukjae Cho, and Seongsoo Hong. Schedulability-aware Mapping of Real-Time Object-Oriented Models to Multi-Threaded Implementations. In *7th International Workshop on Real-Time Computing and Applications Symposium*

(*RTCSA 2000*), 12-14 December 2000, Cheju Island, South Korea, pages 7–14. IEEE Computer Society, 2000.

- [18] Sharath Kodase, Shige Wang, and Kang G. Shin. Transforming Structural Model to Runtime Model of Embedded Software with Real-Time Constraints. In *Design, Automation and Test in Europe Conference and Exposition (DATE 2003)*, 3-7 March 2003, Munich, Germany, pages 20170–20175. IEEE Computer Society, 2003.
- [19] Jamison Masse, Saehwa Kim, and Seongsoo Hong. Tool Set Implementation for Scenario-based Multithreading of UML-RT Models and Experimental Validation. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2003)*, May 27-30, 2003, Toronto, Canada, pages 70–77. IEEE Computer Society, 2003.
- [20] Jeffrey R. Merrick, Shige Wang, Kang G. Shin, Jing Song, and William Milam. Priority Refinement for Dependent Tasks in Large Embedded Real-Time Software. In *11th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'05)*. IEEE Computer Society, 2005.
- [21] Birger Møller-Pedersen. SDL Combined with UML. *Teletronikk*, 4:36–53, 2000.
- [22] OMG. UML Profile for Schedulability, Performance, and Time Specification. *Version 1.0, formal/03-09-01*, September 2003.
- [23] OMG. UML 2.0 Superstructure Specification. *OMG Revised Final Adopted Specification (ptc/04-10-02)*, October 2004.
- [24] OMG. UML Profile for Modelling Quality of Service and Fault Tolerance Characteristics and Mechanisms. *OMG Adopted Specification, ptc/2004-06-01*, June 2004.
- [25] RTL. Software and Tools for Communicating Systems. <http://www.laas.fr/RT-LOTOS/>.
- [26] Hossein Saiedian and Srikrishnan Raguraman. Using UML-Based Rate Monotonic Analysis to Predict Schedulability. *IEEE Computer*, 37(10):56–63, 2004.
- [27] M. Saksena, P. Karvelas, and Y.Wang. Automated Synthesis of Multi-Tasking Implementations from Real-Time Object-Oriented Models. In *3rd International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2000)*, pages 360–367, Newport Beach, CA, USA, March 2000. IEEE Computer Society.
- [28] Manas Saksena and Panagiota Kervelas. Designing for Schedulability: Integrating Schedulability Analysis with Object-Oriented Design. In *The 12th Euromicro Conference on Real-Time Systems*, June 2000.



- [29] B. Selic, G. Gullekson, and P.T. Ward. *Real-Time Object-Oriented Modeling*. John Wiley and Sons, 1994.
- [30] B. Selic and J. Rumbaugh. Using UML for Modeling Complex Real-Time Systems. March 1998. Whitepaper Available from www.objecttime.com.
- [31] OMEGA ST. Project. <http://www-omega.imag.fr/>.
- [32] Telelogic. Telelogic Products. <http://www.telelogic.com/products/>.
- [33] TTool. A Toolkit for Editing and Validating TURTLE Diagrams. <http://www.eurecom.fr/apvrille/TURTLE/index.html>.
- [34] Jing Xu, C.M.Woodside, and D.C.Petriu. Performance Analysis of a Software Design using the UML Profile for Schedulability, Performance and Time. In P.Kemper and W.Sanders, editors, *Proc. of 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation - Performance TOOLS'2003*, volume 2794 of *LNCS*, pages 291–307, Sept 2003.

ABOUT THE AUTHORS



Abdelouahed Gherbi is a PhD student at the Electrical and Computer Engineering Department, Concordia University, Canada. He obtained an Engineer degree in Computer Engineering and a M. Sc. degree in Software Engineering from the University of Constantine, Algeria. His main research interests include UML-based modeling and analysis of embedded and real time systems, the acceleration of Embedded Java Virtual Machines, and the Software Security. He can be reached at gherbi@ece.concordia.ca.



Ferhat Khendek received an Engineer degree in Computer Engineering, option Software, from the University of Tizi-Ouzou, Algeria, in 1986, the M. Sc. and Ph. D. degrees in Computer Science from the University of Montreal in 1989 and 1995 respectively. During his Ph. D studies, he held a research fellowship from the IBM Center for Advanced Studies in Toronto. Currently, Ferhat Khendek is an Associate Professor with the Electrical and Computer Engineering Department at Concordia University. In July 2001, he has been appointed Concordia Research Chair in Telecommunication Software Engineering. During the 2001/2002 academic year he was on a sabbatical leave with Ericsson Research Canada. Ferhat Khendek is a member of l'Ordre des Ingenieurs du Quebec (OIQ), Communication Society, IEEE Computer Society, and the SDL Forum Society. He can be reached at khendek@ece.concordia.ca.