JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2010**

Vol. 9, No. 2, March - April 2010

UML Profiles for Modeling Real-Time Communication Protocols

Barath Kumar and Juergen Jasperneite
inIT - Institute Industrial IT
Ostwestfalen-Lippe University of Applied Sciences, Lemgo, Germany

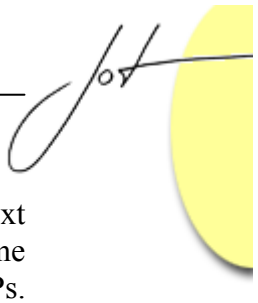
Abstract

Validation of non-functional and functional properties of these protocols during the early stages of design and development is important to reduce cost resulting from protocol anomalies, design errors like deadlock or livelock situations and/or violations of time constraints. The MDD approach promises a well-adapted formalism to bridge real-time protocol complexities and time to market pressure. UML with its several diagrams, supports techniques to overcome the aforementioned complexities. It also supports quantitative analysis through its real-time profiles. This paper reviews the most important real-time UML profiles which can be used for designing time-critical Industrial Communication Protocols (ICPs).

1 INTRODUCTION

Communication systems as we know, serve as the backbone of any distributed system. They specify the chronology of the interaction between the communicating entities. Thus, when it comes to industrial applications, the design and development of ICPs of high reliability is very important because accurate exchange of protocol and application messages in a real-time environment is critical. However, development and design of such high quality industrial protocols for real-time environment is quite difficult and a complex task, as it demands developers with expertise in the field of 'protocol engineering and system design', implementation, validation, optimization and maintenance for developing such protocols. Further, protocols and systems have to be comprehensively designed and tested functionally before determining its performance. Errors in design result in redesign, which is costly and may lead to delays in delivery of the protocols or systems under development, which is not desirable.

The advent of Unified Modeling Language version 2.0 (UML 2.0), promises to be a major breakthrough in the field of formal description techniques. The built-in extensibility mechanisms of UML 2.0 which facilitates extending UML 2.0 through profiles makes UML 2.0 an important candidate for modeling real-time protocols.



Therefore, it is crucial to analyse the capabilities of various UML profiles in the context of modeling time-critical ICPs. Thus, in this paper we explore the capabilities of real-time UML profiles in expressing real-time, functional and non-functional properties of ICPs. The paper is not intended to provide a detailed description of these UML profiles, rather it focuses on showing how the main concepts of these profiles are useful in designing time-critical ICPs.

This paper is organised as follows: section 2 deals with some of the built-in features of UML 2.0 for expressing real-time properties, section 3 is dedicated to UML profile for Schedulability, Performance and Time. The UML profile for Quality of Service (QoS) is explained in section 4. In section 5 we focus on UML CS profile dedicated for Communicating Systems. We review Rational UML profile UML-RT for designing real-time applications in section 6. The recently standarized OMG UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE) is discussed in section 7 and the paper ends with our conclusion in section 8.

2 REAL-TIME FEATURES AVAILABLE IN UML 2.0

The UML 2.0 Specification [OMG05a] contains a rich set of concepts and modeling elements that supports modeling of some aspects of real-time systems. For example some functional characteristics of real-time systems like concurrency, timing constraints, etc. can be modeled using the inbuilt real-time features of UML 2.0.

Concurrency

Concurrency modeling in UML 2.0 is supported by active objects, concurrent operations and concurrent composite states. Further, 'communication diagram' provides message sequencing mechanism to model sequences of events, which is achieved by assigning a sequence number to each message. For e.g. let us consider the following set of messages [KGG+06]:

2a: [Enable=false]*‖ [t=1, 2….n ms] Read (t)

2b: Calculate (t)

In the above example, messages with sequence numbers 2a and 2b indicate that they can be simultaneously passed or triggered, provided that all messages with sequence number 1.x have been successfully triggered or passed already and the guard expression [Enable=false] is satisfied for triggering this message. The symbol '*' shows that the following operation 'Read' is an iterative operation, which is performed every ms and '‖' indicates that there exists at least one or more such operations that executes concurrently along with this operation.

While, Communication diagrams use message sequencing mechanism to model concurrency, Sequence diagrams use the concept of Combined Fragment to indicate parallel execution of set of operations. Combined Fragment, containing the interaction

operator ‘par’ as shown in figure 1 can be used to express concurrent operation. Figure 1 shows an interaction with one server and two users. Each user makes a request; the server performs the service and replies to the user. In this example, there is no specified ordering among the two interaction sequences which, indicates that the request event of the two users may occur concurrently.

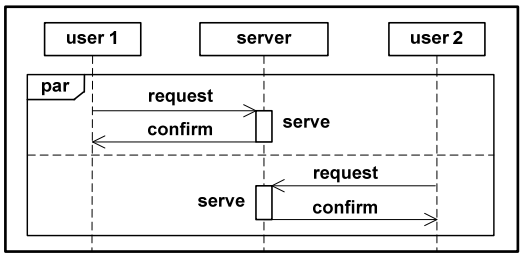


Figure 1: Parallel combined fragment in sequence diagram [RJB06]

Timing constraints

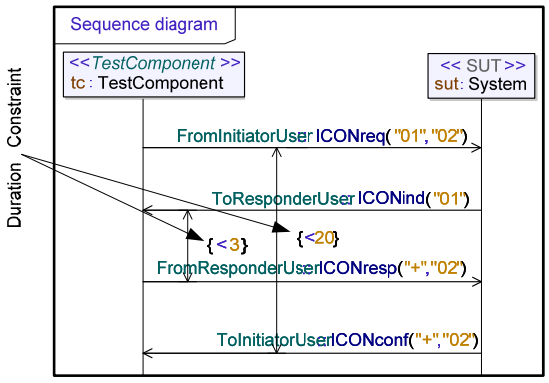


Figure 2: Sequence diagrams showing Time Constraints [KJ08]

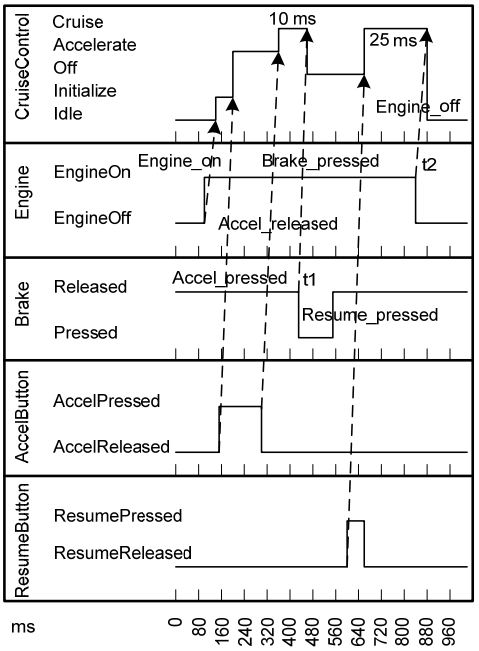
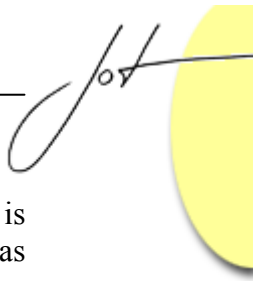


Figure 3: Timing diagram detailing some real-time features [KGG+06]

In addition to concurrency the other important characteristic of real-time ICPs are the ‘Timing constraints’. In order to express timing constraints UML 2.0 provides Time Event concept with the help of two data types namely Time and Time Expression [GK06], [OMG05a] which specifies an absolute or relative (relative to the occurrence of other events) point in time when the event occurs using expressions. These times related



constraints can be used either in state diagrams or in sequence diagrams (the latter is shown in figure 2). The 'Time constraints' can be in the form of duration, as well as certain instants of time; sequence diagram supports both.

In timing diagrams, time is expressed on the linear axis while the message/event interaction among the different lifelines/objects and their effects can be expressed using different states for the lifelines. Figure 3 presents an example of a timing diagram of a Cruise Control system studied in [KGG+06] showing the possible system behavior with time when message/events are passed among different lifelines of the cruise control system. This figure shows, on receiving the brake_pressed and engine_off signals, the cruise control system should satisfy the real-time constraints of 10 ms and 25 ms respectively.

According to a scientific work [KGG+06], the most visible way of expressing time related constraints is provided by Timing diagrams. However, when it comes to modeling real-time constraints, which involves event handling among many lifelines, Sequence diagrams score over Timing diagrams.

UML 2.0's extension mechanism

Apart from the above mentioned inbuilt features of UML 2.0 for modeling real-time ICPs, the most important feature is UML 2.0's built-in extensibility mechanisms; which facilitates extending UML 2.0 through profiles which support real-time modeling. These profiles are discussed in the following sections.

3 UML PROFILE FOR SCHEDULABILITY, PERFORMANCE AND TIME

UML profile for schedulability, Performance and Time (UML/SPT) [OMG05b] is an OMG standard UML profile suitable for modeling and analysis of real-time systems. It is a framework to model resource, time and concurrency concepts, and to support predictive quantitative analysis of UML models by supporting schedulability and performance analysis. The profile does not invent any new techniques, but rather codifies the art of capturing timeliness and related properties. The profile provides stereotypes, tagged values and constraints with specific names that can be used to annotate UML diagrams with quantitative details. The primary benefit of annotating UML models with these information, is the ability to exchange quantitative properties between different tools, such as UML modeling tools and analysis tools for schedulability and performance analysis. Figure 4 shows the expected Paradigm for the SPT Profile. Here in this figure, the Model developer designs the UML model using a UML modeling tool. Once the UML model is designed, the developer annotates the model with quantitative information. Then the annotated UML model is converted into an analysis model appropriate for analysis by the Analysis tool. Upon completion of the analysis, the analysis tool updates the user model which can then be validated.

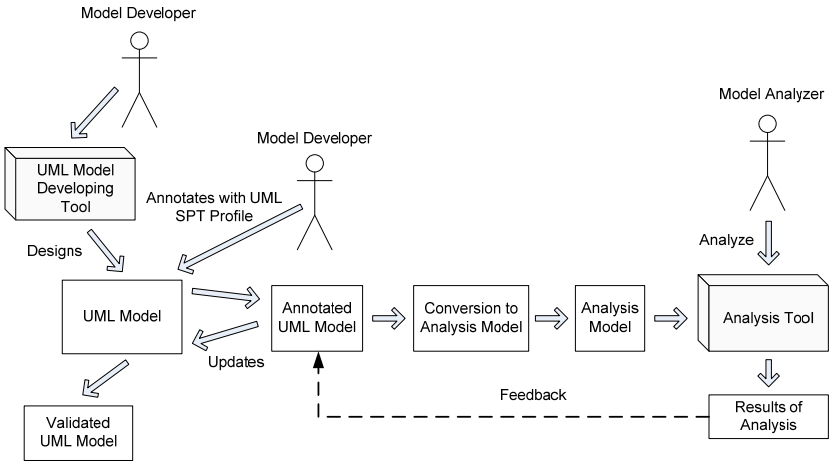


Figure 4: Usage Paradigm for the SPT Profile, based on [Woo07]

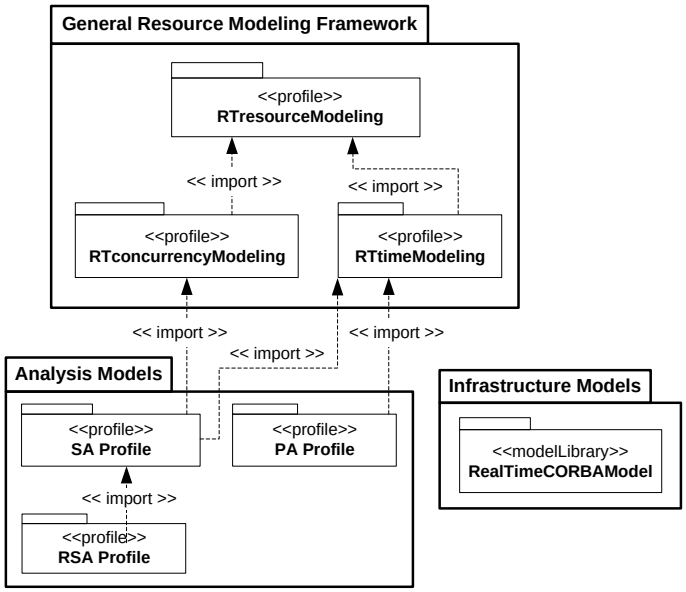


Figure 5: Organizational structure of SPT profile [OMG05b]

The UML SPT profile is divided into 3 sub-profiles for better understandability and usage. The organizational structure of the profile with its three primary packages namely a) General Resource Modeling Framework, b) Analysis Model c) Infrastructure is shown in figure 5.

General Resource Modeling Framework

This package contains the basic concepts of real-time systems. It is further subdivided into 3 more sub-packages or sub-profiles:

- **RT Resource Modeling :** It defines the basic concepts of “resource” and “quality of service”. These are refined and extended progressively as we move down the

profile's structure. A resource is defined as a model element with finite properties like safety, time, availability, capacity etc. Quality of service is defined as a quantitative specification of a limitation on one or more services offered by a resource.

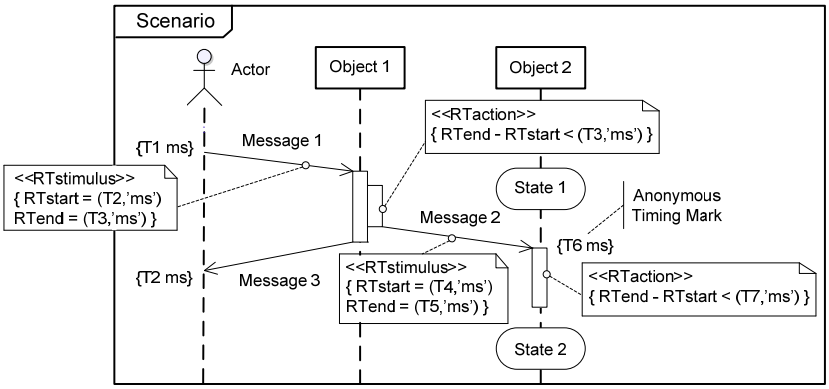


Figure 6: UML Model Annotated with RT Time Modeling sub-profile, based on [Dou06]

Stereotype	Description
<<RTtime>>	to specify time values.
<<RTdelay>>	to specify delay activities.
<<RTinterval>>	to specify time intervals.
<<RTtimer>>	to specify a timer mechanism.
<<RTaction>>	to specify an action that takes time.
<<RTclock>>	to specify a clock mechanism.
<<RTtimeout>>	to specify a timeout action.
<<RTstimulus>>	to specify a timed stimulus.

Tag	Description
RTstart	to specify starting time.
RTend	to specify ending time.
RTperiodic	to specify periodicity of the timer.

Table 1: Commonly used Stereotypes and Tag values of RT Time Modeling

- **RT Time Modeling :** This sub-profile specifies tags and stereotypes for modeling time and time-related concepts. Figure 6 shows a UML diagram annotated with this sub-profile. Table 1. shows the most commonly used stereotypes and associated tagged values provided by this sub-profile. Where, <<RTtime>> can be used to express time values; time constraints can be expressed using <<RTdelay>>, <<RTinterval>> etc. and <<RTtimer>>, <<RTclock>>, <<RTtimeout>> etc. can be used to express time related mechanisms.

- **RT Concurrency Modeling :** This sub-profile refines the vague notations of concurrency defined in core UML, so they can be used efficiently for concurrency modeling. The most important concept of Concurrency Model is the 'Concurrent unit'; it is an active resource instance that executes concurrently with other concurrent units. It is stereotyped as <<CRconcurrent>> and the main thread is represented using a tagged value on the class, referencing a method. Figure 7 depicts a UML diagram annotated with this RT concurrency modeling sub-profile.

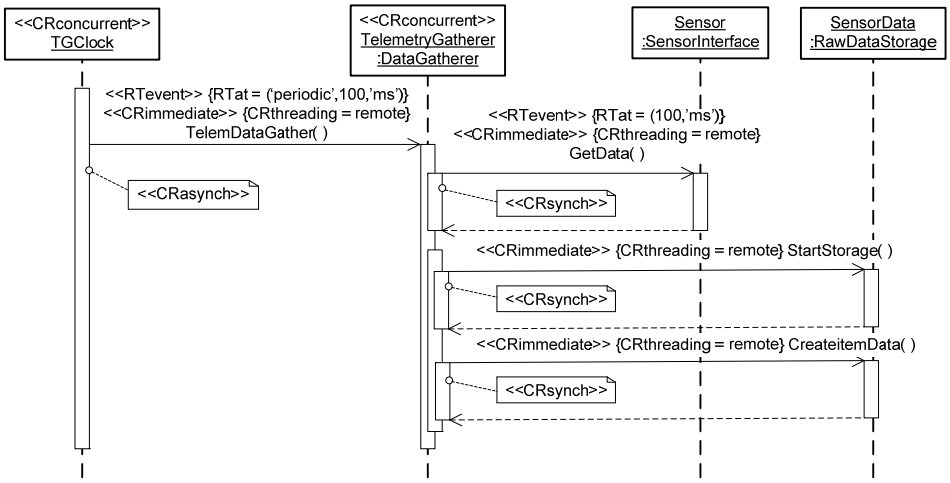


Figure 7: UML Sequence diagram annotated with RT Concurrency Modeling sub-profile [OMG05b]

Stereotype	Description
<<CRconcurrent>>	to define a concurrent unit concept.
<<CRasynch>>	to specify an asynchronous invocation.
<<CRsynch>>	to specify an synchronous invocation.
<<CRimmediate>>	to represent the concept of immediate service instance.

Tag	Description
CRthreading	to describe whether the receiving instance creates its own concurrent execution thread to handle the service request (local) or assumes that there is an existing thread available (remote).
CRmain	to describe the main method of a concurrent unit, such as a thread.

Table 2: Commonly used Stereotypes and Tag values of RT Concurrency Modeling

Analysis Modeling Package

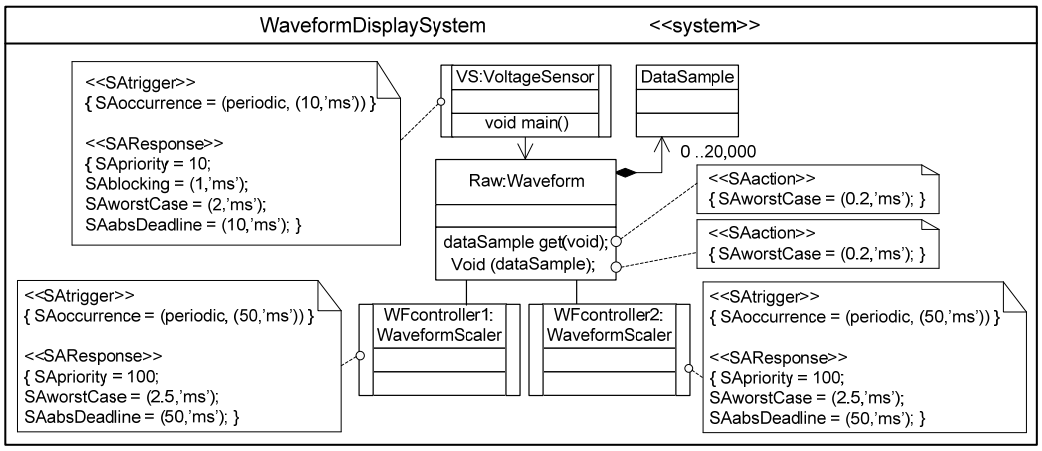


Figure 8: UML Model annotated with 'SA Profile' stereotypes, based on [Dou06]

This package provides sub-packages for different kinds of analysis. It is subdivided into 2 sub-packages or sub-profiles:

- **SA Profile:** This sub-profile deals with schedulability analysis of UML models. It is in this sub-profile where the concepts specified in the aforementioned sub-profiles like general resource model, time and concurrency sub-profiles come together for schedulability analysis. Table 3 shows the most commonly used stereotypes and associated tagged values provided by this sub-profile. Figure 8 shows resources and active objects, with separate threads. The tag values specified along with the constraints attached to these threads, express their properties such as period, occurrence pattern, worst-case execution time, priority etc.

Stereotype	Description
<<SAction>>	to specify a schedulable action.
<<SAResponse>>	to specify a response to a stimulus or action.
<<SAschedulable>>	to specify a schedulable resource that can be used to execute one or a set of actions.
<<SATrigger>>	to indicate a trigger.

Tag	Description
SApriority	for priority specification.
SAblocking	to specify blocking of an action, from execution by a lower-priority task that owns a required resource.
SAabsdeadline	to specify an absolute deadline time instant by which an action should be completed.
SAoccurrence	to specify, how often the trigger occurs. (e.g. periodic).
SAworstCase	to specify the maximum time required for completion of an execution by an action (including overheads such as delay, blocking, etc.).

Table 3: Commonly used Stereotypes and Tag values of SA Profile

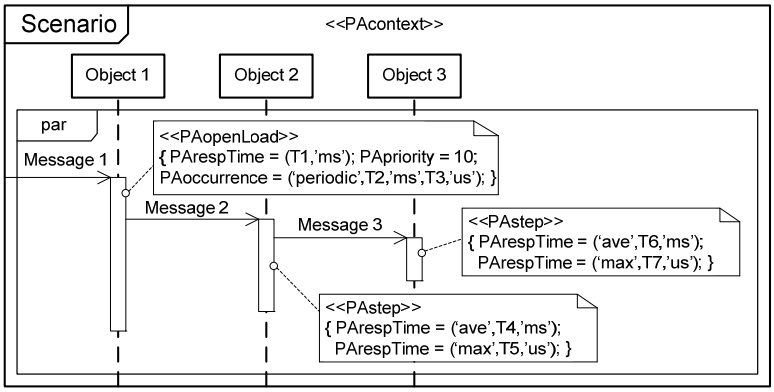


Figure 9: UML diagram annotated Performance Analysis Sub-profile (PA profile), based on [Dou06]

- **PA Profile:** This sub-profile deals with annotation of models for computing the performance of the system. The stereotypes and tags provided by this profile can be used for including quantitative measures for performance analysis. The PA

profile refines the concepts and tags specified in the aforementioned sub-profiles, for performance analysis. Figure 9 shows a UML diagram annotated with stereotypes of PA sub-profile, it should be noted that the executions on the various processors are modeled in parallel regions using 'par' operator of sequence diagram (combined fragment).

Stereotype	Description
<<PAhost>>	to specify an execution engine that hosts the scenario.
<<PAopenLoad>>	to specify an open workload.
<<PAstep>>	for specifying a step in a scenario.
<<PAcontext>>	to indicate a performance analysis context.

Tag	Description
PArespTime	to specify the time required for completion of the scenario from the starting time of the scenario.
PAoccurrence	to specify the workload's arrival pattern.
PApriority	to specify the priority of a workload.
PArep	to specify the number of repetitions of a step.

Table 4: Commonly used Stereotypes and Tag values of PA Profile

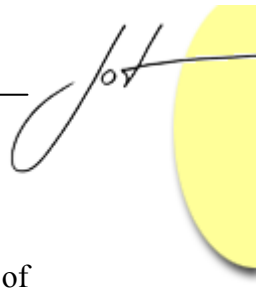
Real-Time CORBA (Common Request Broker Architecture) Sub-profile (Infrastructure Modeling)

The main objective behind introducing this sub-profile in UML SPT profile is to provide an extension to facilitate schedulability analysis of Real-Time CORBA applications. It is a middleware standard owned by OMG. Since this sub-profile falls beyond the scope of our research work, we are not discussing this sub-profile in detail. Interested readers can refer to [OMG05b] for more information.

UML SPT profile is too vast to be completely covered by this paper. Hence, to have an exhaustive understanding of this profile, interested readers can refer to [GK06], [OMG05b], [Woo07] and [Dou06].

4 UML PROFILE FOR QUALITY OF SERVICE (QOS)

The QoS profile was adopted by OMG in June 2004 [OMG08]. The QoS profile provides the user with facilities to define a wider variety of QoS requirements and properties as compared to SPT profile, which focuses mainly on schedulability and performance analysis [BP04]. The SPT profile was the first attempt to extend UML with real-time properties like timing and concurrency concepts, for expressing requirements and properties needed for performing schedulability and performance analysis. The QoS profile on the other hand was introduced to complement UML/SPT profile. It broadens the scope of specifying real-time properties by allowing the users to define open variety of quality of service requirements and properties [BP04]. UML/QoS allows model developers to define any set of quality of service requirements and perform specific analysis like performance, schedulability and even dependability [Mig03].



Proposed annotation procedure of QoS profile

The annotation procedure proposed by the Quality of Service (QoS) profile consists of the following three steps:

- **Definition of QoS characteristics:** As a first step, the QoS characteristics of interest for the analysis to be carried out on a specific system/protocol domain are defined. These QoS characteristics are template classes having parameters that have to be instantiated in the next step. The QoS profile allows users to define new QoS characteristics which leverage, through specialization, the general QoS Characteristics catalogue defined in the QoS profile. This feature of UML/QoS, paves way to a natural way of customizing the QoS characteristics. In this context the QoS profile is more flexible as compared to SPT profile.
- **Definition of Quality Model:** The QoS characteristics (having parameters), which were defined in the previous step for annotation of the UML model, should be assigned actual values. Quality characteristics bound class definition and template binding can be used for this purpose. The UML model, with the binding information and the bound classes is called the Quality Model.
- **Annotation using QoS constraints**: In this step, the UML models are annotated with the specified QoS constraints and QoS values related to the QoS characteristics defined in the earlier steps. As per the QoS profile, there are three possible ways to annotate the UML model, they are as follows:

    1. By attaching a note symbol with a QoS constraint written in OCL to a constrained model element.
    2. By connecting the constrained model element with an instance of class stereotyped as ‘QoS Value’
    3. Or, by stereotyping the constrained model element with a QoS constraint and use ‘AllowedValue’ and ‘LogicalOperator’ properties to reference a set of QoS values and their logical relationships.

SPT profile Vs QoS profile

The major differences between these two annotation techniques are: a) the QoS profile annotation technique is not as straight forward as the annotation technique of SPT profile, and b) the QoS profile technique is more flexible as it supports customization of QoS characteristics. The SPT Profile does not allow model developers to customize its stereotypes and tags. The set of stereotypes and related tags of SPT profile can only be used without customization to annotate model elements for a given type of analysis.

5 UML PROFILE FOR COMMUNICATING SYSTEMS

UML 2.0 is a collection of several semi-formal standard notations and concepts for modeling software systems at different stages and views of the same system. The lack of

strong formality in non-profiled UML is beneficial at the early stages of development. In later stages of simulation, validation and implementation, UML is too imprecise to fulfill this task for which SDL is well suited. While UML 2.0 features multiple viewpoints, informal object models and property model views, SDL offers detailed formalized object models with respect to execution semantics. However, in practice, UML is made more formal by binding semantic variations in the UML language and providing a more precise behavior either in the form of a tool or as a language profile [WKH06].

As per [WKH06], the goal of "UML profile for Communicating Systems" (UML CS) [Wer06] is to bridge the gap between the requirement, analysis and design phase by combining the strengths of UML and SDL. The UML CS profile by means of stereotypes, tagged values and constraints has extended the elements of composite structure diagrams, class diagrams and state machines in such a way that their usage becomes more obvious and aligned to SDL. Apart from providing a mapping to SDL, the profile provides additional set of high-level modeling concepts and language elements, which are not currently supported by SDL, but are necessary for modeling and engineering current and upcoming communication protocols. From Industrial Communication Protocols' point of view, they can be used in 'Virtual Automation Networks' (VAN) [VAN08].

Concept of the UML CS profile

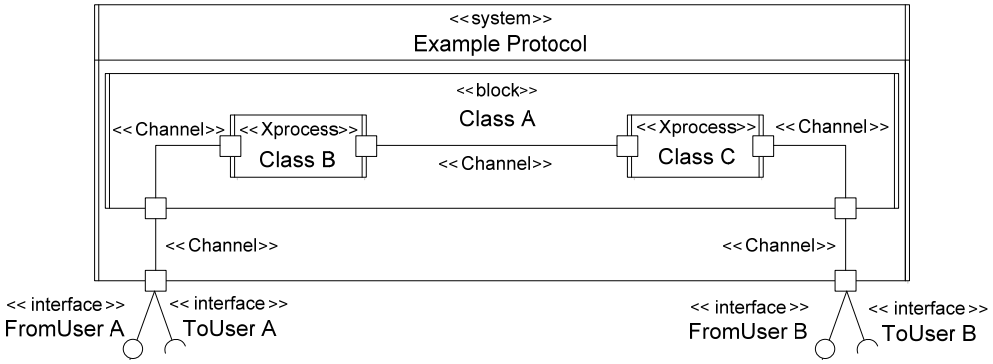
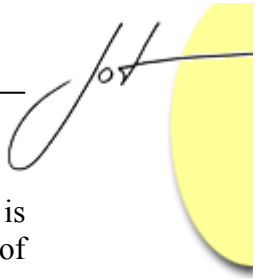


Figure 10: Composite Structure diagram of an Exmple Protocol Using UML CS Profile

The structural modeling elements of UML CS profile are similar to SDL, i.e. it supports system, block, process and package agents. Decomposition of system design is supported by nesting agents within other agents. A system's structure can be specified using classes and composite structures. Classes can be used to define the types of agents (system, block or process) which can then be instantiated with a composite structure. Further, operations, attributes, signals and timers with their scopes can also be defined. The composite structure instantiates the agents along with their multiplicities and specifies the communication paths by means of connectors or channels. It also specifies interfaces for signals and remote procedure calls (figure 10 shows an example of usage of UML CS profile). Agents modeled by active classes which execute a behavior after initiations are called 'Processes'. State machines are used for describing the behavior of processes. The



only exception is, only activities can be used to define data type operators. This is because an activity in UML CS must not wait for triggers. Further, an additional set of stereotypes provided by UML CS profile supports specifying ASN.1 based data types. However, only a subset of the ASN.1 notations is supported. To have an exhaustive understanding of the UML CS profile, interested readers can refer to [WKH06], [Wer06] and [Woc06].

Major extensions to UML 2.0 for Real-time Communication Protocols

- **Random:** For simulation of real-time communication protocols, it is very important to simulate situations which arise by chance, for e.g. frame loss. UML CS has operations which can generate random values for various distributions like Poisson, Erlang, etc. CallBehaviorAction is extended by the <<random>> stereotype to provide operations which generates pseudo-random values when executed.
- **Time:** The time model of UML 2.0 has a very limited functionality. With this simple time model, it is possible to start a timer in a state but it involves waiting for it to trigger the corresponding transition. This simple time model feature of UML 2.0 does not cover the full needs of ICPs unlike SDL's time model, which is very well suited for this purpose. Like SDL's time model, UML CS time model is well suited for ICPs, as the starting of a timer is independent of waiting for the timeout trigger. The 'Timeout' tag can be used to represent the time when the timeout event should occur.

6 UML PROFILE FOR REAL-TIME SYSTEMS (UML RT)

UML RT [SR98] is a profile which extends the basic UML concepts to facilitate the design of complex real-time systems [Sel98]. UML RT is a modeling language by itself, which allows model developers to design event-driven, complex and even distributed real-time systems. UML-RT's modeling notations have their origin from real-time specific modeling language ROOM [SGM+92].

UML-RT is an industrial standard [AT05], which primarily focuses on architecture specification of real-time systems/protocols. Further, the behavior of the system can be modeled using UML state machine diagrams. UML-RT has introduced four new building blocks (namely capsules, ports, connectors and protocols) to the standard UML meta-model for modeling the structure and the behavior of complex real-time systems.

Structural Modeling

The three new structural constructs introduced by UML-RT for structural modeling are capsules, ports and connectors:

- **Capsules:** These are active architectural objects that model complex software components. Capsules normally have associated state machines that can process

(send and receive) messages via their ports. Capsules communicate with its surrounding and sub-capsules via 'Ports'. Sub-capsules and their connections can be used to describe the internal structure of the components or capsules.

- **Ports:** Ports mediate the interaction of the capsule with its surrounding. Ports connected to a state machine of a capsule (end port) can handle messages sent to them. On the other hand, ports connected to ports of sub-capsules, simply behave as forwarding ports (relay ports) which forward signals to sub-components.
- **Connectors:** These are communication channels which can interconnect only ports and provide the transmission facilities necessary for supporting a particular protocol. It should be noted that ports can only interconnect ports that realize complementary roles of their mutual protocol.

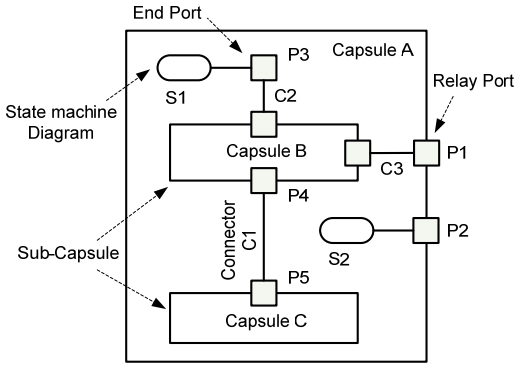


Figure 11: Component Architecture of UML RT

Figure 11 shows UML-RT component architecture that includes the aforementioned concepts, where Capsule A is made up of sub-capsules Capsule B and C. Port P1 (relay port) of Capsule A is connected to a sub-capsule Capsule B, which owns an internal port P3 (end port). Port P3 connects the port of sub-capsule Capsule B, with a state machine diagram. Similarly, Port P2 (end port) of Capsule A is connected to a state machine diagram which specifies the behavior of Capsule A. Connectors are used for modeling communication channels between two or more ports. Ports P3 and P5 realize complementary roles of their mutual protocol and are connected with the help of a Connector C1. The systems behavior at the architectural level is defined by its 'Protocols' and 'Connectors'.

Behavioral Modeling

State machines specify the functional behavior of the components. Capsules without state machines are only containers for sub-capsules. Compared to UML state machines, UML-RT state machines do not allow concurrent states. Signals arriving at end ports are handled by state machines. 'Protocol' is a new construct introduced by UML-RT with respect to behavior modeling. Protocols specify a desired behavior that can occur over a connector.

Even though, UML-RT provides constructs to model complex real-time industrial protocols, the major pitfall of UML-RT is, its modeling entities and syntactic constructions lack precise semantics and clearly defined syntax. Further, UML-RT does not support modeling timing issues. However, when it comes to scheduling, the profile introduces a set of common scheduling annotations (including absolute and relative deadlines, worst-case completion time and priority) which are fairly sufficient to perform basic schedulability analysis. To have a detailed understanding of UML RT, interested readers can refer [SR98], [Sel98], [SGM+92], [AT05] and [Sel05].

7 UML MARTE PROFILE

The recently standardized UML Profile for Modeling and Analysis of Real-Time and Embedded systems (UML MARTE Profile) [OMG07] introduces domain specific ideas relevant for modeling real-time and embedded systems design involving the ability to express non-functional properties, execution platforms, resource allocation, quantifiable notions of time, etc. available within the unified modeling framework. These features of UML MARTE, from our project's point of view are very important because modeling of ICPs involve a great deal of the above mentioned non-functional runtime properties.

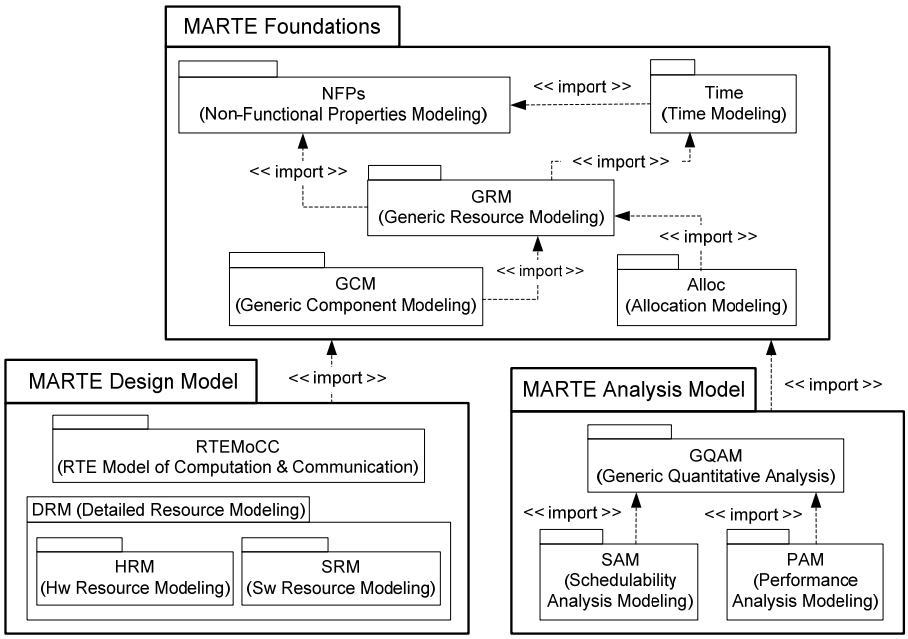


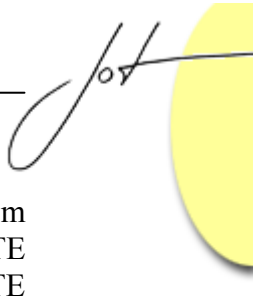
Figure 12: Organizational Structure of MARTE Profile, based on [DTA+08]

The MARTE profile is intended to replace the existing UML Profile for Schedulability, Performance and Time (SPT) [OMG05b]. The architecture of MARTE profile consists of three main packages namely MARTE Foundation package, MARTE Design Model and MARTE Analysis Model as shown in figure 12.

MARTE Foundation package

This package defines all the basic foundational concepts required for design and analysis of real-time and embedded system. It provides model developers with constructs for modeling the following:

- **Non-functional Properties (NFPs):** This sub-package offers means for specifying the non-functional properties of the real-time system like memory usage, power consumption, etc. It also explains how these NFPs can be attached to the model elements. In short, this sub-profile specifies a general framework for annotating UML model elements with NFPs.
- **Time Modeling:** This package enables time and time-related structure modeling. The main objective of MARTE is to provide basic and advanced time modeling concepts. As a profile succeeding the UML SPT profile, MARTE has to support a metric time with implicit reference to physical time [AMS07]. However, MARTE supports general time models like 'physical', 'logical' and multiform. These advanced time related concepts could then be used to develop various Models of Computation and Communication (MoCC).
- **Generic Resource Modeling (GRM):** This Sub-package provides all the necessary stereotypes and tagged values to represent resources like communication media, computing resources, storage resources etc. Further, it includes features that are needed for dealing with modeling of executing platforms with different levels of abstraction and modeling of both 'hardware' (e.g. physical communication channels) and 'software' (e.g., real-time operating systems) platforms. The GRM package along with Time modeling package can be used for specifying timing constraints and when used with the NFP package can be used for specifying the quality of services.
- **Generic Component Model (GCM):** This package is useful for applying component base strategies in the domain of real-time and embedded systems. GCM package is nothing but refinements applied to the structured classes of UML on top of which a support SysML blocks has been added. Thus this model provides a common denominator for various models of computation and communication (MoCCs) [OMG07]. This package serves to provide a model as general as possible, that is independent of a specific execution semantics, upon which real-time characteristics can be applied at a later point in time.
- **Allocation Modeling:** The MARTE profile allows designers to model both applications and execution platforms. An application element in MARTE could be a service, a computation or a real-time operating system (RTOS) function. An execution platform is a set of connected resources representing the hardware architecture. It consists of <<HW_ Resource>> such as storage resource (RAM, ROM or Cache), communication devices (Bus and I/O devices) and computing resource (processor, hardware accelerator) etc. After modeling the application and hardware architecture it is important to map the application tasks onto the execution platform in real-time embedded applications. Mapping places an



important role when it comes to performance. MARTE provides a mechanism called 'allocation' which allows designers to specify mapping. A MARTE allocation is an association between a MARTE application and a MARTE execution platform. The main concept of allocation is <<Allocate>>, it is used for associating elements from a logical context, application model elements, to named elements described in a more physical context, execution platform model elements [BMP+07].

The concepts defined using this foundation package are then refined in the following two packages to support modeling and analysis concerns of the real-time system.

MARTE Design Model

This package addresses model-based design starting from requirement capture to specification, design and implementation. It provides high level concepts for modeling both, quantitative and qualitative features of real-time systems/protocols. Further, it also provides means for detailed description of software and hardware resources used for execution of an application.

- **RTE Model of Computation and Communication (RTEMoCC):** The main objective of this package is to provide high-level modeling concepts, that enable modeling quantitative features like deadline and period, and qualitative features that are related to behavior, like communication and concurrency.
- **Detailed Resource Modeling (DRM):** This sub-package specializes generic concepts provided by the GRM sub-package. It offers specific modeling artifacts for specifying both, software and hardware execution supports. The DRM package consists of Software Resource Modeling (SRM) and Hardware Resource Modeling (HRM) package.
- **Software Resource Modeling (SRM):** This package specifies a set of modeling artifacts that facilitates describing the structure of execution supports like RTOS (Real-time operating system). RTOS is used as execution support in multi-tasking-based approach, which is the widespread approach used for designing software real-time applications. The SRM package does not intend to define a new API (Application programming interface) for RTOS rather it offers modeling artifacts to model libraries of RTOS API.
- **Hardware Resource Modeling (HRM):** This package is used to specify the detailed platform architecture elements i.e., its purpose is to describe hardware execution supports with different levels of details and views that are essential to fulfill the application specification. The HRM package is intended to serve description of existing hardware and conception of new hardware platforms. The HRM consists of two views, a logical view and a physical view; these two views are complementary to each other. They provide abstraction of hardware in two different ways that could be simply merged. The physical view classifies hardware resources based on physical properties on the one hand

and on the other hand, the logical view classifies hardware resources based on functional properties [OMG07].

Once the Hardware Resource Modeling and Software Resource Modeling are completed, they are combined to support the whole application execution.

MARTE Analysis Model:

This package offers specific abstractions and relevant annotations that could be read by analysis tools. MARTE analysis is intended to provide trustworthy and accurate evaluations using formal quantitative analysis based on sound mathematical models [OMG07]. This package is sub-divided into three other packages, namely:

- **Generic Quantitative Analysis Modeling (GQAM):** This sub-package provides generic concepts for quantitative analysis. Generic Modeling provides basic modeling concepts and NFPs based on NFP annotation framework, which simplifies the package and makes it easier for the designers to add new analysis based on their requirement. The GQAM sub-package offers specialized domains, whose analyses are based on software behavior, like schedulability and performance and also other NFPs like power, security, availability, memory and reliability. In other words, we can say that the main objective of GQAM domain is to reveal the resource usage based on system behavior. This feature of MARTE Analysis makes us believe that MARTE will play an important role when it comes to modeling of ICPs. The constructs defined using this sub-package are then refined in the other packages for schedulability and performance analysis.
- **Schedulability Analysis Modeling (SAM):** This sub-profile deals with schedulability analysis. It helps predicting, whether the system under study as the ability to meet certain temporal constraints, like miss rations, deadlines, etc. Scheduling influences timing and performance in a crucial manner when it comes to real-time applications like ICPs and hence schedulability analysis becomes important for calculating guaranteed bounds on response times and resource processing loads. This sub-package provides a group of common annotations for schedulability analysis.
- **Performance Analysis Modeling (PAM):** The sub-package offers support for performance analysis, i.e. it provides ways to determine whether a system can provide adequate performance under non-deterministic behavioral conditions, based on statistical values. The PAM package includes both, ‘single-case’ analysis, for a particular set of given input parameter and also ‘multi-case’ analysis, for e.g. ‘sensitivity analysis’ which helps in identifying risky workload situations, ideal operational parameter, etc. and ‘capacity analysis’ which identifies the design or configuration capabilities.

For a very detailed understanding of UML profile for MARTE, readers are requested to refer [OMG07], [DTA+08], [FBS+07], and [HH08].

8 CONCLUSION

The complexity of real-time Industrial Communication Protocols demand usage of adequate modeling language. In our previous research work [KJ08], we were able to show that UML 2.0 can be applied to all phases of industrial communication protocol engineering aided with automated testing facility. However, in [KJ08] we concentrated more on functional properties, but non-functional real-time properties are as important as functional properties. Hence, this work reviews the various real-time UML 2.0 profiles which can be used for engineering non-functional real-time properties of ICPs.

Based on our review, we can say that UML SPT profile has initiated many research works in the context of addressing non-functional properties. Nevertheless, UML SPT does have a considerable amount of limitations as discussed in [Ger04], [OMG] and [WP04]. The UML QoS profile, which was introduced to complement UML SPT profile, is more flexible as it allows users to customize the QoS characteristics but lags when it comes to supporting symbolic variables and expressions. In [BP04], it is explained, that both SPT and QoS profiles struggle to cope with the balance between flexibility and simplicity/convenience of expression. The UML MARTE profile, which succeeds the UML SPT profile uses model based approach together with quantitative analysis approach (SAM and PAM sub-profiles). This allows model designers to annotate additional information needed for various analysis to be attached directly on to the actual design model, rather than creating dedicated models for analysis. Thus, we have planned to investigate the potential of these features of UML profile for MARTE for engineering non-functional real-time properties of Industrial Communication Protocols.

REFERENCES

[AMS07] C. André, F. Mallet and R. de Simone: "Time Modeling in MARTE", *in proceeding of Forum on specification & Design Languages (FDL '07)*, Barcelona, Spain, September 2007.

[AT05] K. B. Akhlaki and M. I. C. Tunon: "Combining the Description Features of UML-RT and CSP+T Specifications Applied to a Complete Design of Real-Time Systems", *in proceeding of World Academy of Science, Engineering and Technology*, vol. 7, Prague, Czech Republic, 26 -28 August, 2005.

[BMP+07] P. Boulet, P. Marquet, E. Piel, and J. Taillard: "Repetitive Allocation Modeling with MARTE,"*in proceeding of Forum on specification and design languages (FDL'07)*, invited paper, Barcelona, Spain, September 2007.

[BP04] S. Bernardi and D. C. Petriu: "Comparing two UML Profiles for Non-functional Requirement Annotations: the SPT and QoS Profiles" *in proceeding of Second International Workshop on Specification and Validation of UML Models for Real Time and Embedded Systems (SVERTS)*, Lisbon, Portugal, 11 October 2004.

[Dou06] B. P. Douglass: *Real Time UML Third Edition*, Addison Wesley, Feb 2006.

[DTA+08] S. Demathieu, F. Thomas, C. André, S. Gérard and F. Terrier: "First Experiments Using the UML Profile for MARTE", *in proceeding of 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC 2008),* pp. 50-57, USA, 5-7 May 2008.

[FBS+07] M. Faugère, T. Bourbeau, R. de Simone and S. Gérard: "MARTE: Also an UML Profile for Modeling AADL Applications", *in proceeding of 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*, pp. 359-364, Auckland, New Zealand, 11-14 July, 2007.

[Ger04] S. Gerard: "Report on SIVOES 2004 – SPT", *in proceeding of Workshop on the usage of the UML profile for Scheduling, Performance and Time*, Toronto, Canada, 2004.

[GK06] Abdelouahed Gherbi and Ferhat Khendek:"UML Profiles for Real-Time Systems and their Application", *in Journal of Object Technology*, vol.5, no.4, May-June 2006, pages 149-169, http://www.jot.fm/issues/issues_2006_05/article5

[HH08] M. Hagner and M. Huhn: "Tool Support for a Scheduling Analysis View", *in proceeding of Design, Automation and Test in Europe (DATE'08)*, Munich, Germany, 10-14 March 2008.

[KGG+06] M. U. Khan, K. Geihs, F. Gutbrodt, P. Gohner and R. Trauter: "Model-Driven Development of Real-Time Systems with UML 2.0 and C", *in proceedings of the 4th Workshop on Model-Based Development of Computer-Based Systems and 3rd International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, March 2006.

[KJ08] Barath Kumar and Jürgen Jasperneite: "Industrial Communication Protocol Engineering using UML 2.0: a Case Study", *in 7th IEEE International Workshop on Factory Communication Systems (WFCS 2008),* Dresden, Germany, May 2008.

[OMG] OMG: "Pending Issues sent to the OMG Finalization Task Force: UML Schedulability, Performance and Time Profile", http://www.omg.org/issues/uml-scheduling-ftf.open.html.

[OMG05a] OMG: "Unified Modeling Language: Superstructure", Version 2.0, formal/05-07-04, August 2005.

[OMG05b] OMG: "UML Profile for Schedulability, Performance, and Time Specification", OMG Adopted Specification, Version 1.1, formal/05-01-02, January 2005.

[OMG07] OMG: "A UML Profile for MARTE", OMG Adopted Specification, Version Beta 1, OMG Document Number: ptc/07-08-04, August 2007.

[OMG08] OMG: "UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms", Version 1.1, formal/2008-04-05, April 2008.

[RJB06] J. Rumbaugh, I. Jacobson and G. Booch: *The Unified Modeling Language Reference Manual, Second Edition*, Addison Wesley, February 2006.

[Sel98] B. Selic: *Using UML for modeling complex real-time systems*, Lecture Notes in Computer Science 1474, Springer-Verlag, pp. 250-260, 1998.

[Sel01] B. Selic: "A UML Profile for Modeling Complex Real-Time Architectures", Rational Software Inc., 2001.

[SGM+92] B. Selic, G. Gullekson, J. McGee and I. Engelberg: "ROOM: An object-oriented methodology for developing real-time systems", *in proceeding of 5th International Workshop on Computer-Aided Software Engineering*, pp. 230-240, July 1992.

[SR98] B.Selic and J.Rumbaugh: *UML for modeling complex real-time systems,* Technical report, Object Time, 1998.

[VAN08] VAN: "Investigation for future communication", Available at: http://www.van-eu.eu/ , July 2008.

[Wer06] C. Werner: *A UML Profile for Communicating Systems*, PhD thesis, Georg-August-Universitat, Gottingen, Germany, 2006.

[WKH06] C. Werner, S. Kraatz and D. Hogrefe: "A UML Profile for Communicating Systems", *in proceeding of the Fifth Workshop on System Analysis and Modelling (SAM 06)*, pp 81-90, Kaiserlautern, Germany, June 2006.

[Woc06] J. Woch: *A UML Profile for Communicating Systems – Informal semantic Comparison and Example*, Master thesis, Georg-August-Universitat, Gottingen, Germany, November 2006.

[Woo07] M. Woodside: "From Annotated Software Designs (UML SPT/MARTE) to Model Formalisms", *in 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2007*, Bertinoro, Italy, May *28*-June 2, 2007.

[WP04] M. Woodside and D. Petriu: "Capabilities of the UML Profile for Schedulability Performance and Time (SPT)", *in proceeding of Workshop on the usage of the UML Profile for Scheduling, Performance and Time (SIVOES-SPT)*, Toronto, Canada, June 2004.

About the authors

Barath Kumar is a scientific research assistant at Institute Industrial IT (inIT), Germany. He obtained a Bachelor degree in Electrical and Electronics Engineering (B.E) from University of Madras (India) in May 2004. In January 2008 he received Master of Science (M.Sc) degree in Information Technology (Joint International Program in co-operation with three European Universities namely Halmstad University in Sweden, Hochschule Ostwestfalen-Lippe in Germany and Aalborg University Esbjerg in Denmark). His main research interests include engineering of Industrial Communication Protocols using UML 2.0 and timed automata, schedulability and performance analysis of protocols and protocol testing. He can be reached at barath.kumar@hs-owl.de

Juergen Jasperneite is a professor for Computer Networks at the Ostwestfalen-Lippe University of Applied Sciences and the director of Institute Industrial IT (inIT). He studied electrical engineering and received his doctorate degree (Dr.-Ing.) in Electrical Engineering and Information Technology from the Otto-von-Guericke-University of Magdeburg, Germany, in 2002. His professional career included various engineering and management positions. From 1988 – 1989 he was with Bosch Telecom, Berlin, as a R&D-Engineer in the field of GSM technology. From 1989 to 2005 he was with Phoenix Contact, Germany, in different positions, starting as an ASIC-Developer for industrial communication systems and finally as the Head of the R&D-Department of the Business Unit Automation Systems. He is an IEEE Senior Member since 2006 and Author/Co-Author of more than 90 technical papers. He served as a member or reviewer for a lot of scientific conferences and journals. His current research interests include modelling, testing and evaluating of Realtime Communication Systems (Protocol Engineering), especially in the field of industrial automation systems. He can be reached at juergen.jasperneite@hs-owl.de