A UML/MARTE-based design pattern for semi-partitioned scheduling analysis

Amina Magdich*, Yessine Hadj Kacem*, Adel Mahfoudhi*, and Mickaël Kerboeuf†
*University of Sfax, ENIS, CES Laboratory, Soukra km 3,5 B.P: 1173-3000 Sfax, Tunisia
Email: amina.magdich, yessine.hadjkacem, adel.mahfoudhi@ceslab.org
†University of Brest (France), Lab-STICC, MOCS Team
Email: kerboeuf@univ-brest.fr

***Abstract*—The scheduling of Real-Time Embedded Systems (RTES) is a challenging step that requires vast knowledge and expertise about the domain, which makes difficult the step of complex systems scheduling modeling. This paper presents a design pattern intended to support and facilitate the scheduling modeling of multiprocessor systems. The contribution of this pattern is that is designed to i) support semi-partitioned scheduling allowing tasks migration ii) model all the tasks features/types and criteria of scheduling in the same view (only one pattern is used) iii) specify the system properties using a high-level modeling language UML/MARTE (Modeling and Analysis of Real-time and Embedded systems).**

***Keywords*-semi-partitioned scheduling; scheduling algorithms; scheduling analysis; design pattern; UML/MARTE.**

I. Introduction

RTES are integrating more and more functionalities satisfying the users' requirements. This requires the increase in the amount of computing resources to improve the performance of executions. Thereby, it is beneficial to use a multiprocessor architecture, but that makes the scheduling step difficult.

What is worthwhile to note is that three multiprocessor scheduling approaches are available in the literature [10][15]; the partitioned approach, the semi-partitioned approach and the global approach. Regarding the partitioned approach, it affects each task to be executed on one processor. Accordingly, tasks are not allowed to migrate between processors. CPU utilization is therefore not optimal. As for the global approach, it enables a full migration of tasks such that every task may be allocated ,not simultaneously, on different processors. Although the full migration allows reaching optimality, it may cost in terms of context switching. With respect to the the semi-partitioned scheduling approach, it enables a controlled tasks migration. It reduces the number of migrations and then offers a compromise between the number of migrations and the processors occupation. It will be the adopted approach in this paper.

In fact, the increasing complexity of RTES and the difficulty of their scheduling has led designers to use techniques that improve the software process quality, reduce the development time and cost, improve the reuse of models, etc. The use of high-level modeling methods is considered to be a technique fulfilling these constraints. In this context, a design pattern [6] is an appropriate solution to address these requirements. In fact, it facilitates complex systems modeling, allows reuse of models, reduces modeling and maintenance costs and improves software process quality. The design pattern to propose in the present paper is intended to be used for systems scheduling. It represents a generic illustration supporting the modeling of features to be used while scheduling multiprocessor systems using the semi-partitioned approach. The proposed design pattern will be modeled and annotated using UML/MARTE profile.

The remainder of this paper is structured as follows. Section 2 emphasizes the various related works. In section 3, the UML/MARTE profile is defined and the importance of its use in the context of design pattern modeling is shown. Section 4 gives an overview of the proposed design pattern and presents the corresponding view. Section 5 exposes a case study of the suggested design pattern application. In section 6, we discuss and evaluate our proposal. Finally, section 7 provides a summary and conclusions.

II. Related Works

Different research studies have focalized on the use of design patterns to be exploited in different fields. In [16], the authors have suggested design patterns that support the modeling of structural and behavioral properties of concurrent systems and Real-Time databases. These patterns are modeled using UML/MARTE and especially the subprofiles NFP (Non-Functional Properties), Alloc (Allocation Modeling) and HLAM (High-Level Application Modeling). Other studies have been focused on the use of patterns to overcome concurrent systems complexity such as in [3]. In [3], the authors have proposed architectural design patterns dealing with concurrency, resources, distribution and security. In [4], the authors have suggested a design pattern meant to be used in the context of a Design Space Exploration step (DSE) allowing to reduce the time of the DSE models built. To achieve this, they used UML to represent the pattern view and POOSL (Parallel Object-Oriented Specification Language) for pattern mathematic definition. The proposed pattern allows the tasks modeling, the target architecture and the Software/Hardware binding. They will also be exploited as templates that may be applied in different situations.

RTES are facing a set of modeling difficulties among

which the tasks scheduling can be stated. In this context, different research works pertaining to the conception of design patterns to be used in a scheduling context have been carried out. In [5], a thread-based design pattern intended to be exploited in the context of RTES is presented. This pattern, integrated in a scheduling methodology, allows the automatic generation of scheduling, but it is used only for simple systems having small sizes. In [9], a design pattern of an adaptive scheduling for dynamic environments has been elaborated to encapsulate the code associated with multiple tasks versions. The modeling of this pattern is based on UML. What is worthy to mention is that tasks migration was not dealt with in their work. Since the choice of feasibility tests is considered as a hard step for engineers, in [8], the authors have proposed a design pattern-based methodology to support the automatic choice of scheduling algorithms. Their approach integrates five design patterns based on AADL (Architecture Analysis and Design Language) such that each pattern is intended for a specialized use. The approach proposed in this paper is based on the compliance of the AADL architecture to the design pattern. In the same vein, in [2], the authors have explained how to check automatically this compliance test. In [8] and [2], researchers have suggested design patterns that are used to deal only with a partitioned approach and there is no consideration of all the tasks types (periodic/sporadic, dependent/independent, etc). Besides dealing with the scheduling issue, designers have to use five design patterns. To validate their design patterns, they have made use of the Cheddar ADL. In these two papers, the Cheddar tool which does not support semi-partitioned and global scheduling is used for tasks scheduling. It is in this context that the authors have proposed, in [17], an extension for Cheddar to support multiprocessor scheduling algorithms: partitioned and global scheduling algorithms. In [7], the authors have mentioned the process for design patterns composition as they explained with details the five design patterns used throughout their approach. «Ravenscar» pattern is used to deal with communication and synchronization between tasks and access to shared resources. It enables asynchronous task communication using mutex or semaphores. Besides, «Ravenscar» supports only static and off-line scheduling. «Time-triggered communications» deals with task communication based on a shared but not protected memory, and supports only independent tasks. As regards «Blackboard», it allows only the use of reader/writer communication protocol and supports only periodic tasks. «Queued buffer» implements the communication based on the producer/consumer protocol and supports only periodic tasks. «Unplugged» models only periodic and independent tasks.

In the present paper, we contribute by the proposition of a design pattern intended to undergo the semi-partitioned scheduling that allows controlled tasks migration. Our intuitive idea is to propose only one design pattern, based on UML/MARTE [11], containing all the data needed for semi-partitioned scheduling and supporting all the tasks types. This facilitates the modeling step and the future compliance test. In fact, it is easier to check the compliance of a model according to one pattern than checking it according to various patterns. The use of MARTE, which is gathering a big set of annotations, makes designers able to model all the scheduling features in only one pattern encompassing software and hardware criteria. Moreover, MARTE is a graphical modeling language, so it facilitates the modeling of patterns compared to textual modeling languages.

III. UML/MARTE AND SCHEDULING ANALYSIS

A. System Model for scheduling analysis

This section proposes a definition of systems to be modeled through the proposed design pattern to deal with multiprocessor scheduling allowing controlled tasks migration. We consider the semi-partitioned scheduling of complex systems composed of n tasks T=$\{T_1, T_2, ..., T_n\}$ to be allocated upon m processors P=$\{P_1, P_2, ..., P_m\}$. Each task is characterized by four principal parameters (C_i,D_i,P_i,R_i) where C_i is the worst execution time, D_i is the deadline and R_i is the activation date. If a task is periodic, P_i will represent its period, and in case of sporadic tasks, P_i will specify the inter-arrival time between two consecutive releases of a task T_i.

It is important to mention that in case of periodic tasks, all the parameters C_i, D_i, P_i and R_i are known before the application startup (i.e. D_i=critical_delay+R_i). For sporadic tasks, the activation date is to be predicted during the future behaviour of the application. So, for a sporadic task, only C_i and the critical delay must be known before the application launching. Besides, the deadline and the period must be computed after detecting the activation dates. For aperiodic tasks, only C_i is known before the application starting.

B. UML/MARTE capabilities for scheduling analysis

MARTE is a specification of UML (Unified Modeling Language) providing support for the specification, modeling, and early verification of RTES. It offers a big set of subprofiles, stereotypes and attributes facilitating the systems modeling and properties specification.

During the specification of design patterns, various criteria must be considered. Indeed, in the context of scheduling we must take into account all the tasks features and types, tasks communications, tasks/processors communications, tasks states, access to shared resources, etc. Indeed, MARTE enables designers to include all these specifications in the same model (pattern). The proposed pattern must take into consideration the specification of features linked to the semi-partitioned scheduling approach such as the controlled tasks migration between processors.

Originally, MARTE supports only the modeling of the systems to be scheduled according to the partitioned approach.

In the same vein, [14] and [13] have proposed extensions for MARTE profile to enable the modeling of features to be used within the migration context. Our pattern is based on the use of these extensions to incorporate the tasks migration features in the modeling view.

IV. Proposed design pattern

In this section, we describe the importance of the proposed design pattern employment in a scheduling context.

A. Real-Time scheduling pattern specification

The modeling of RTES for scheduling stage is a critical step, which may be overcome using UML/MARTE-based design pattern intended for a semi-partitioned scheduling. The proposed pattern is represented through two UML/MARTE views; a static and dynamic views. Nevertheless, in this paper we will only expose the static view.

1) Overview: A pattern is aimed to support the modeling of complex systems, and may be used to design different systems in the same field without ever being adopted twice similarly. Every pattern must deal with a well-defined problematic. It represents a solution that solves an issue and supports the modeling of complex systems. In the context of semi-partitioned scheduling, the proposed pattern must support all the features that may be required by semi-partitioned scheduling algorithms.

2) Context: The intention of our scheduling design pattern is to help designers to specify all the properties to be used in the context of a semi-partitioned scheduling approach allowing controlled tasks migration.

3) Problem: Scheduling patterns existing in the literature are intended to be used in the context of partitioned scheduling inhibiting tasks migration. Besides, there is no pattern that supports all the tasks types and all the scheduling criteria (communication between tasks, tasks/processor, and access to shared data or resource) in the same view.

4) Solution: The purpose of the suggested pattern is to support semi-partitioned scheduling and all the tasks types, it specifies all the criteria needed for semi-partitioned scheduling in the same view. The pattern modeling is based on UML/MARTE as a high-level modeling language providing a big set of stereotypes to annotate the built views.

B. UML view of the proposed design pattern

The present design pattern is modeled through a static view annotated with MARTE (Figure1). It represents a modeling of the different entities with their relationships as it specifies the corresponding properties that are needed for semi-partitioned scheduling. This view is annotated especially through NFP, SRM (Software Resource Modeling), HRM (HardWare Resource Modeling) and Alloc.

The static view of the proposed design pattern is composed of a set of entities defining Software and Hardware items, such as Tasks, execution hosts (processors), schedulers, mutual exclusion resources, communication means, memories and batteries. A task annotated «swSchedulableResource» is characterized by its name, type (e.g. periodic), deadline, period (i.e. for periodic tasks, we use the attribute «period», otherwise this attribute represents the minimal duration between two activation dates of a sporadic task), laxity and number of its jobs (resMult) (i.e. A task may be divided into different jobs).

A big set of attributes is offered by MARTE to annotate tasks. For example, the attribute «isStaticSchedulingFeature» is used to explore if the scheduling parameters (e.g. priority, deadline, etc) of a task are static or dynamic. The attribute «isPreemptable» specifies if the task may be preemptable by another concurrent resource that has a higher priority. To mention the task activation number allowed in the system, we use the attribute «activationCapacity».

In a semi-partitioned scheduling context, a task may migrate between processors. We use the attribute «P_Host» to identify on which processor a task can be allocated. This is to restrict the task migration to all the processors in order to optimize the scheduling cost. The execution time of a task depends on the processor on which a task is allocated. This will be represented by the attribute «P_execT».

For a semi-partitioned scheduling, we need to use two types of schedulers: a global scheduler and schedulers associated with processors. The global scheduler annotated, in the task class, «host:scheduler» affects tasks to be scheduled through the other schedulers. As a task/job can migrate in a semi-partitioned scheduling, between execution hosts, some attributes may change following the switching context (e.g. deadline, period, scheduler, processor, execution times in the context of heterogeneous processors) and then we take advantage of a multiplicity of [0..*] for all these attributes (for more details see [14][13]).

A computing resource on which a task may be allocated for its execution is represented by the entity Execution Host annotated through «hwComputingResource» and «scheduler». Different types of computing resources are available, such as processor «hwProcessor» or ASIC «hwAsic». The entity Execution Host specifies both the computing resource and the scheduler that is associated with. The entity Execution Host is characterized by its name, utilization, number of cores, etc. The scheduler annotated «scheduler» is defined by a set of attributes from which we specify the host («host: ComputingResource[0..1]») on which a scheduler is running, its speed factor, the set of schedulable resources that are waiting to be scheduled, etc.

In the semi-partitioned context, designers have to use a global scheduler and other schedulers associated with processors. A global scheduler annotated «GaExecHost» is used to partition tasks on the different schedulers. The entity ExclRes annotated «SwMutualExclusionResource» specifies the mutual exclusion resources that are used to protect the concurrent access to shared resources. Such entity is marked by an integer value. The communication between

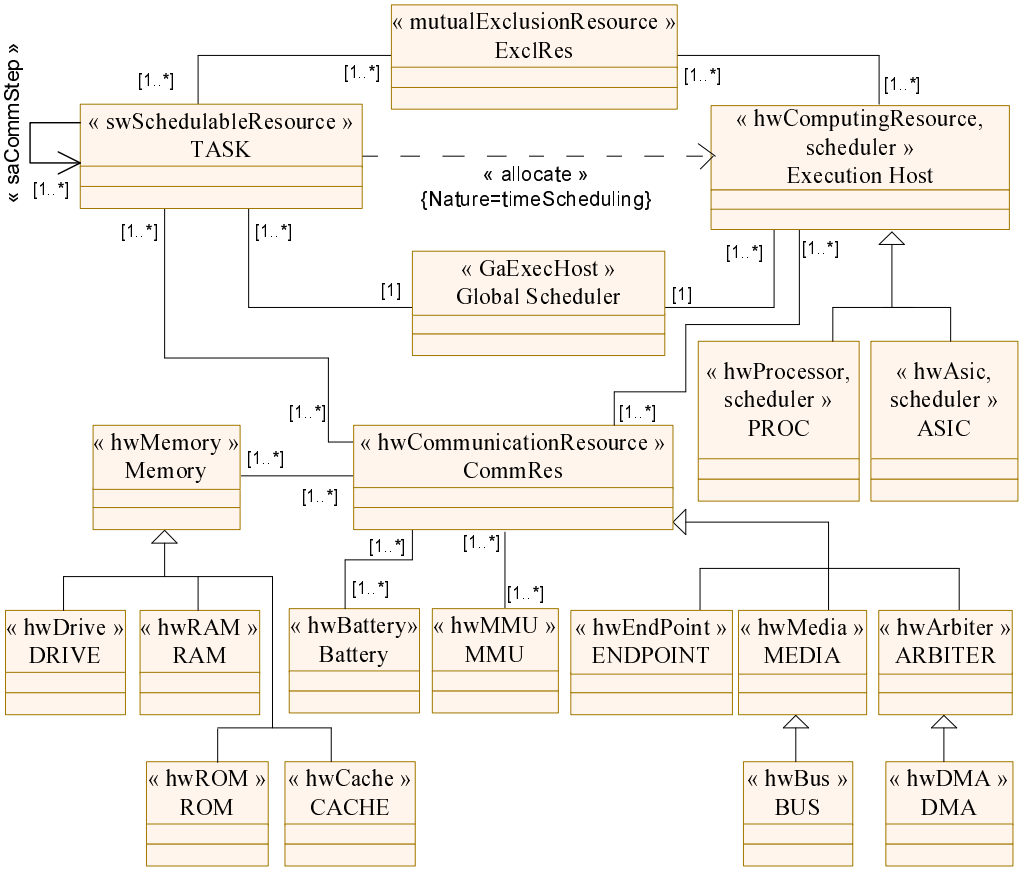

Figure 1: Static view of the proposed scheduling design pattern

the different components of an RTES is established through a communication resource «hwCommunicationResource» which may be a bus of communication «hwBus», Direct Memory Access «DMA», end point «hwEndPoint» (e.g. antenna) or a memory «hwMemory» (e.g. RAM, ROM).

V. Case study

To evaluate our proposal, we provide an example of systems to be modeled using the proposed design pattern. The same example is used in [15] to validate a semi-partitioned scheduling algorithm. The parameters used in this example are those of Marvell's XScale technology-based embedded processor PXA270 [1].

The studied system is used to be simple and helpful to clarify the proposed design pattern. The system is composed of ten real-time, periodic and independent tasks as well as an architecture target including four identical processors, a communication bus, two memories, a battery and of course schedulers. Each processor is protected by a mutual exclusion resource («mutualExclusionResource» class). The concurrence between the tasks to lock a shared resource is better illustrated using a dynamic view. In the static pattern, the concurrence is illustrated using associations and operations (annotated through stereotypes and attributes). In this example, tasks are independent so that there is no send or receive of data between them. Nevertheless, the communication between tasks may be represented in a static view using an association linking tasks annotated through «saCommStep». Under the stereotype «saCommStep», a big set of attributes is available to specify the criteria of communication. It is to be noted that this system allows only the migration of tasks and prohibits that of jobs. To handle semi-partitioned scheduling, the studied system needs one global scheduler running on an independent execution host

Task Name	Ri	Ci	Di	Pi
T1	0	6	20	20
T2	0	6	15	15
T3	0	13	40	40
T4	0	15	40	40
T5	0	6	30	30
T6	0	12	20	20
T7	0	8	20	20
T8	0	10	25	25
T9	0	6	10	10
T10	0	8	20	20

Table I: Tasks parameters

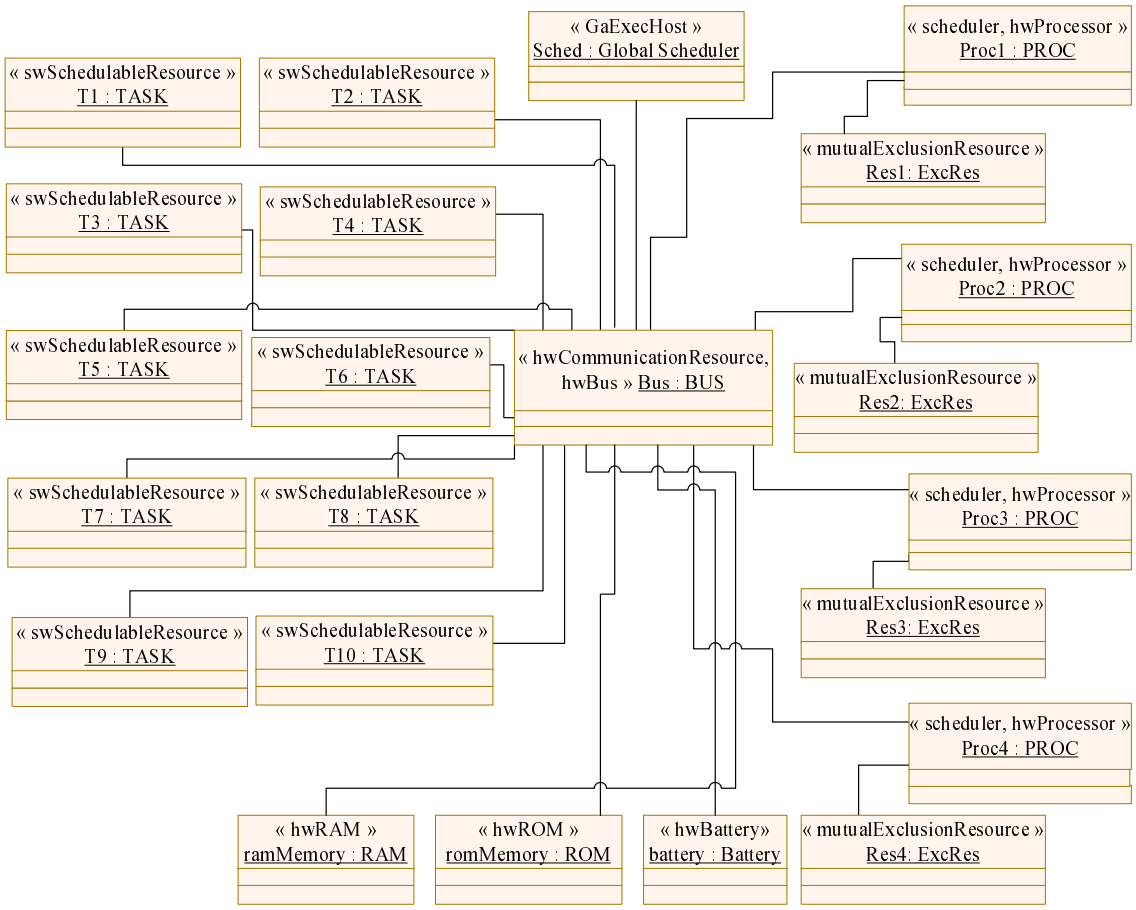

Figure 2: An example of the proposed pattern reuse

and four schedulers associated with processors.

Table I specifies the different tasks parameters that are needed for scheduling. In fact, R_i represents the activation date of every task, C_i is the Worst execution time, D_i represents the deadline and P_i is the corresponding period. These parameters must be entered in the «swSchedulableResource» classes to guide the scheduling step.

Figure 2 specifies an application of the proposed pattern and specifies the different tasks and the target architecture.

As specified before, task T1 is periodic, which is indicated through the attribute «type: ArrivalPattern» that is set on «periodic». The possible processors to which T1 can migrate are P1 and P2. The estimated execution times are specified using «P_execT» attribute. The required scheduling type is static, which is mentioned by setting the value of «isStaticSchedulingFeature» on «true».

VI. Discussion and evaluation

Although design patterns currently exist in the literature to support scheduling, they are used to handle monoprocessor or partitioned scheduling inhibiting tasks migration. They are designed through textual and graphical languages such as AADL or UML profile. In this paper, we proposed a design pattern used to support semi-partitioned scheduling enabling tasks migration. Its design was done through UML/MARTE profile. The use of MARTE has facilitated the designing step since it is a high-level modeling language supporting a big set of stereotypes and attributes. Another contribution of the present paper is the use of only one pattern, rather than five [7], to support semi-partitioned scheduling. The use of only one pattern, including all the scheduling criteria, makes simple the compliance test between the pattern and a model. In fact, it is easier to check the compliance between the proposed pattern and a model than between many patterns and a model (for example, in [2] to check the compliance of a model, five compliance tests must be done. In our proposal only one compliance test is used to check the compliance of a model to a pattern).

For the evaluation of the suggested design pattern, we aim to validate the two major issues: the applicability of the design pattern on systems that may be scheduled using the semi-partitioned scheduling approach and the importance of the use of MARTE profile within the context of design patterns modeling. In the case study elaborated in section 4, we have

designed a static pattern instance that models a multiprocessor system allowing tasks migration. This instance has shown the ability of the suggested design pattern to support semi-partitioned scheduling (restricted tasks migration) and multiprocessor architectures. It has also shown the ability of the pattern to specify any tasks parameters and criteria. The specification of tasks or architecture parameters was easy to do due to the use of MARTE profile that includes a big set of attributes. The case study investigated in this paper has also shown that in only one pattern, it is possible to specify all the criteria needed for the scheduling analysis step.

VII. Conclusions and future works

This paper proposes a design pattern providing support for multiprocessor RTES semi-partitioned scheduling modeling. This pattern was modeled using UML and annotated via the MARTE profile. It was represented through a static view. A case study has been performed to validate the use of our pattern in a scheduling modeling context.
The proposed pattern may be used to deal with ordinary scheduling analysis (check the schedul ability of a system) by mapping the features of the MARTE view to a scheduling tool such as in [12]. It may also be exploited to endure an automatic choice of semi-partitioned scheduling algorithms. Indeed, the choice of the appropriate algorithm to schedule RTES is a critical task for engineers. Consequently, we will seek to integrate the proposed design pattern in a methodology enduring the automation of the suitable semi-partitioned algorithm choice for the multiprocessor systems.

References

[1] Marvell's xscale microarchitecture. http://www.marvell.com/.

[2] Pierre Disseaux, Alain Plantec, Mickael Kerboeuf, and Frank Singhoff. AADL design patterns and tools for modelling and performance analysis of real-time systems. In *5th european congress ERTSS Embedded Real-Time Software and System.*, France, May 2010.

[3] Bruce Powell Douglass. *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[4] Oana Florescu, Jeroen Voeten, Marcel Verhoef, and Henk Corporaal. Reusing real-time systems design experience through modelling patterns. In *FDL*, pages 375–381, Darmstadt, Germany, September 19-22 2006. ECSI.

[5] René Fritzsche, Christian Ristig, and Christian Siemers. An approach and design pattern for intra-application scheduling. Technical Report IfI-10-11, Clausthal University of Technology, 2010.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Professional, Boston, MA, USA, 1 edition, 1995.

[7] Vincent Gaudel, Frank Singhoff, Alain Plantec, Pierre Dissaux, and Jérôme Legrand. Composition of design patterns : from the modeling of rtos synchronization tools to schedulability analysis. In *EWiLi'13, The 3rd Embedded Operating Systems Workshop*, Toulouse, France, August 26-27 2013. ACM SIGBED.

[8] Vincent Gaudel, Frank Singhoff, Alain Plantec, Stephane Rubini, Pierre Dissaux, and Jerome Legrand. An ada design pattern recognition tool for aadl performance analysis. In *Proceedings of the 2011 Annual International Conference on Special Interest Group on the Ada Programming Language*, SIGAda '11, pages 61–68, Denver, Colorado, USA, 2011. ACM.

[9] Rodrigo Gonçalves, Rômulo Silva de Oliveira, and Carlos Montez. Design pattern for the adaptive scheduling of real-time tasks with multiple versions in rtsj. In *SCCC*, pages 65–73. IEEE Computer Society, November 2005.

[10] Joël Goossens. Introduction à l'ordonnancement temps réel multiprocesseur. In *Ecole d'été Temps Réel*, pages 157–166, 2007.

[11] OMG Object Management Group. A uml profile for marte: Modeling and analysis of real-time embedded systems. standard, June 2008.

[12] Yessine HadjKacem, Adel Mahfoudhi, Amina Magdich, Walid Karamti, and Mohamed Abid. Using mde and priority time petri nets for the schedulability analysis of embedded systems modeled by uml activity diagrams. In *ECBS*, pages 316–323, April, 2012.

[13] Amina Magdich, Yessine Hadj Kacem, and Adel Mahfoudhi. Extending uml/marte-grm for integrating tasks migrations in class diagrams. In Roger Y. Lee, editor, *SERA (selected papers)*, volume 496 of *Studies in Computational Intelligence*, pages 73–84. Springer, 2013.

[14] Amina Magdich, Yessine Hadj Kacem, Adel Mahfoudhi, and Mohamed Abid. A MARTE extension for global scheduling analysis of multiprocessor systems. In *the 23th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pages 371–379. IEEE Computer Society, November 2012.

[15] Bhatti Muhammad Khurram, Belleudy Cécile, and Auguin Michel. Two-level hierarchical scheduling algorithm for real-time multiprocessor systems. *JSW*, 6(11):2308–2320, 2011.

[16] Saoussen Rekhis, Nadia Bouassida, Rafik Bouaziz, Claude Duvallet, and Bruno SADEG;. Modeling real-time applications with reusable design patterns, 2010.

[17] Stephane Rubini, Christian Fotsing, Frank Singhoff, Hai Nam Tran, and Pierre Dissaux. Scheduling analysis from architectural models of embedded multi-processor systems. *EWiLi Workshop*, 2013.