A MARTE-Based Design Pattern for Adaptive Real-Time Embedded Systems

Ahmed Ben Mansour, Mohamed Naija and Samir Ben Ahmed
El Manar University, Faculty of Mathematical, Physical and Natural Sciences of Tunis Manar, Tunisia

Keywords: Adaptability, Real-time & Embedded Systems, MDE, MARTE, Design Patterns.

Abstract: The design of adaptive real-time & embedded systems (RTES) is a hard task due to the complexity of both software/hardware features and the variability in the operational environment. This paper presents a new design pattern intended to support and facilitate the co-modeling and scheduling analysis of RTES. The contribution of this pattern is that is designed to i) support scheduling analysis allowing adaptation mechanisms ii) model all the software/hardware features and allocation in the same view iii) annotate the system with functional and non-functional properties using a high-level modeling language. The generated model from the pattern contains all the needed information to perform the schedulability tests. As a proof of concepts, we present experimental results for an automobile system.

1 INTRODUCTION

The modeling of Real-Time Embedded Systems (RTES) may be stated as a crucial problem in the software engineering domain with the current lack of design models and tools. RTES are subject to a multitude of constraints (e.g., battery, temperature) and real-time requirements, and have to execute in a highly variable environment (e.g., update available, hardware crash, position change). To retain the system non-functional properties (NFPs) (e.g., deadlines), RTES must be able to adapt themselves to the changes in their operating environment and context. Lightening the task of adaptive systems designers and reducing the development cost and time to market represents a significant challenge in the field, which requires the use of high-level approaches such as MDE (Model Driven Engineering) (Schmidt, 2006) and MARTE (Modeling and analysis of real-time embedded systems) (OMG, 2008).

MDE is a way to beat the growing complexity of adaptive systems by allowing reuse, portability, and automation of the design models. In particular, pattern-based development is becoming an increasingly recognized solution in software engineering by addressing new challenges to support the whole lifecycle co-design of complex systems. In the RTES domain, its adoption is seen promising for several purposes: allowing the reuse of models (Rohnert and Stal, 1996), improving software process quality and reducing the development cost of complex systems.

Design pattern development is a relatively old discipline that has been proven to be beneficial for getting fast and reusable designs (Gamma, 1995). Unfortunately, most of the patterns that are devoted to adaptive systems are expressed as informal indications on how to solve some adaptive mechanisms. These patterns do not include sufficient semantic descriptions, including those of adaptability concepts, for automated processing and to extend their use. For that, MDE can provide a solid basis for formulating design patterns that can incorporate adaptability concepts at several levels of abstraction.

In the present paper, we propose a design pattern intended to guide the designer in the modeling of adaptive real-time embedded systems. It represents a generic illustration backing the co-modeling of features to be used in the context of modeling adaptability and analysis time constraint in the real-time embedded systems. This work is the result of previous investigations and published work. Starting from (Naija et al., 2015) a MARTE-based approach was proposed to concurrency model construction at early design stages. Notably, we perceive a MARTE semantics limitation in modeling level. Accordingly, we have identified the needed of the new version of the MARTE profile supporting adaptation mechanisms. In (Naija et al., 2016) and (Naija and Ahmed, 2016a) we have proposed several extensions in the UML MARTE profile for supporting adaptation mechanisms. In this paper, we present how to use these extensions in a new design pattern.

Our contribution is to be integrated into a model-based approach to guide RTES designers for building and analysing adaptive RTES models. It facilitates complex systems modeling, decreases the development time and cost and increases software process quality. The above benefits have been demonstrated through the application of our proposition to an adaptive system.

The remainder of this paper is organized as follow. In Section II, we give background on related concepts to the present article. Section III provides a survey of some similar works. In Section IV, we present an overview of the proposed design pattern. Section V submits a case study application of the suggested design pattern. Finally, this paper is concluded with some outlined directions for future works.

2 THEORETICAL BACKGROUND

In this section, we present enough information about UML MARTE profile and adaptability concepts to understand how and why we use them in our design pattern.

2.1 UML/MARTE

The UML/MARTE profile (Modeling and Analysis of Real-Time and Embedded systems) (OMG, 2008) defines a framework for annotating functional and non-functional properties of embedded systems. MARTE fosters the building of models that support the specification of scheduling analysis problem. This profile can model tasks, dependencies between them and events under shape a Workload Behavior and platform of the system.

In particular, the Core Elements sub-profile of MARTE Foundations package defines the elements needed to specify configurations, modal behavior and their semantics of execution, as shown in Figure 1.

The modal behavior is strongly related to the notion of mode, which can represent a state of the system in which it must provide a set of features. The mode is the abstract representation of a set of functionalities afforded by a system or a subsystem. When adapting to new operational requirements, a system may have (i) to change from a source mode to a target mode, and (ii) to improve the software application configuration (e.g., by disabling or enabling communication links between components). This design model is used at runtime.

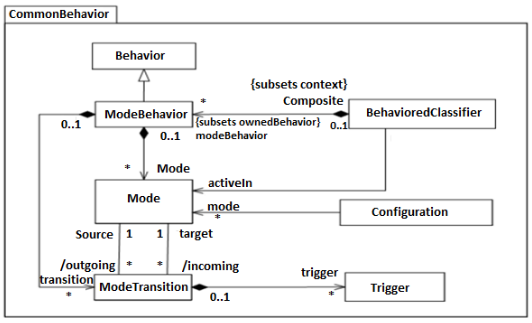

Figure 1: Modal behavior model of MARTE (OMG, 2008).

However, mutually exclusive modes are described from the ModeBehavior model to clarify that only one mode between them can be triggered at the same time. Besides, an operational mode is characterized by a given configuration. This configuration is defined by the allocation of all the elements of the software architecture to the platform components. The transition mode describes the mode change of a system. The latter is invoked in response to an event that marks a change in the execution context. Various criteria must be considered in the context of adaptive real-time & embedded systems during the specification of the design pattern. Indeed, we must take into consideration the different tasks features, execution modes, and tasks/process execution, etc. MARTE does not enable designers to encapsulate all these specifications in the same model.

2.2 Adaptability

There are several definitions of adaptation in the literature. In our previous work (Naija et al., 2016), we defined adaptation as any modification in the structure, behavior or architecture of the system to accommodate the external or internal change of their operating environment or context and according to predefined adaptation plan and rules (Naija and Ahmed, 2016b). In the adaptive system, additional information has to be modeled such as adaptation rules, context, transitional modes and adaptation plan to specify both functional and non-functional properties. In the following, we explain, the concepts of context, rules and adaptation plan:

- **Context:** annotate the minimal representation of the operational environment which is attached to the functional mode.
- **Rules:** specify how the system should be adapted to the new environment requirements.
- **Adaptation Plan:** is the process of taking action to manage or reduce the consequences of a changed configuration.

3 RELATED WORKS

The design of adaptive RTES is relatively old domain and thus very rich. Unfortunately, this task presents a challenge due to the complexity of the problem it handles (Said et al., 2014). In this paper, we focus on approaches and design flows that mainly deal with high-level design and verification of adaptive systems, which are still not well tackled.

In (Said et al., 2014), the authors have proposed five patterns representing generic modeling of the adaptation loop modules. The patterns take into consideration the real-time features of adaptation operation. These design patterns are presented in the static view through class diagrams annotated with MARTE profile stereotyped. The adaptation is then considered as a dynamic and partial change of the operation mode without supporting platform resources.

In the same vein, (Ramirez and Cheng, 2010) have proposed several patterns to design adaptation process. However, the authors introduced adaptation-oriented design patterns to support monitoring, decision-making, and reconfiguration of adaptive systems. They extend the pattern templates used by Gamma et al. (Gamma, 1995) for description design patterns with Behavioral and Constraints fields. Unfortunately, real-time constraints allowing the scheduling test, which is an ad-hoc test, are not modeled.

The authors have proposed in (Supriana et al., 2018), an approach through requirements modeling languages directed to adaptation pattern. The model is prepared through contextual conditions approach that is integrated into MAPE-k (monitor analyze, plan, execute - knowledge) pattern in goal-oriented requirements engineering.

The author in (Hamerski et al., 2018) has introduced a middleware to support the development of self-adaptive management services and applications for MPSoCs. The proposed middleware is heavily based on object-oriented practices and design patterns to present platform abstraction, modular design, and decoupled distributed programming model. However, the author discusses many import features of real-time embedded systems such as performance and time constraint without taking into consideration the interaction between the different components.

In (Abuseta and Swesi, 2015) a MAPE loop (Monitor, Analyzer, Plan and Execute) design process is proposed. However, the authors discuss the interaction between the different components of the loop without taking into consideration the essential features of RTES, such as time constraint, scheduling resource, etc.

Other efforts have been specifically tailored to the partitioned and the global approaches. In (Magdich et al., 2014) and (Magdich et al., 2015), (Magdich et al., 2018), the authors proposed a new UML/MARTE-based design patterns to supporting semi-partitioned and global scheduling. The advantage of these patterns is the automatic selection of the scheduling algorithm and approach. The beneficial of all the earlier mentioned works is they can facilitate the design of adaptive real-time systems. They tackled only one specific adaptation mechanism, which consequently compromises their reusability to adapt the system to new system requirements and constraints. They assume that adaptability is a dynamic change in the task allocation without taking into account operating mode adaptation which is an essential ax of adaptation, unlike this approach. In this paper, we suggest a design pattern providing support several adaptability mechanisms and real-time constraint

4 PROPOSED DESIGN PATTERN

4.1 Real-Time Adaptive Pattern Specification

In this section, we present the description of our pattern. We follow the pattern templates presented in (Buschmann et al., 2007) for our description. Four fundamental fields were detailed in this paper, with are: Overview, Context, Problem, Solution. We use UML MARTE based design pattern intended for adaptability. Our pattern will be limited only to the static view for UML MARTE.

1. Overview: Pattern is aimed to support the modeling of adaptive real-time 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 problem. It represents a solution that solves an issue and supports the modeling of complex systems at the early design stage.
2. Context: Our design pattern intends to guide designers to specify all the properties to be used in the context of an adaptive RTES allowing controlled mode transitions.
3. Problem: MARTE provides support the modeling, specification and early verification of RTES, unfortunately, it does not provide a solution for modeling adaptability.
4. Solution: The purpose of the suggested pattern is to support the modeling of software/hardware ar-

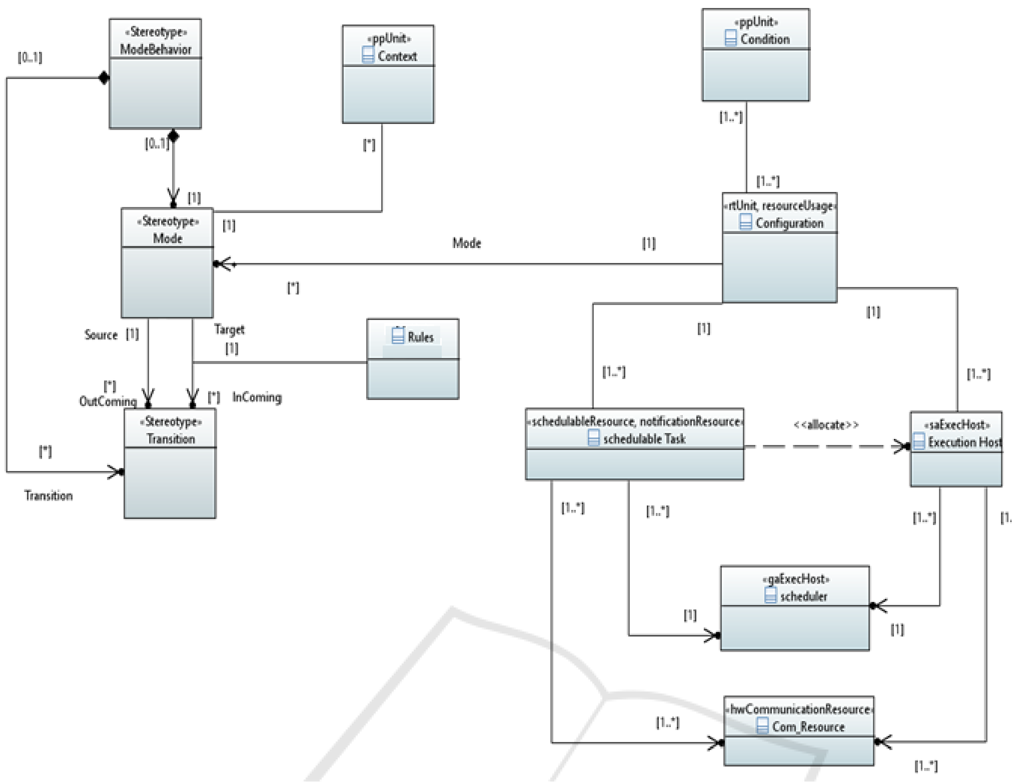

Figure 2: Static view of the proposed adaptive design pattern.

chitecture of the system with adaptive constraints in the same view. The pattern modeling is based on UML/MARTE as a high-level modeling language that provides a big set of stereotypes to annotate the built views.

4.2 Proposed Design Pattern

Adaptation means changing the functionality of the system to accommodate a change in the execution context. Our pattern is based on the MARTE *CoreElement* sub-profile, we propose to add some classes to the base model to be compatible with the requirements of adaptive systems. Then, we use some stereotype of MARTE to support the configuration context.

As indicated in the *CommonBehavior* package, the change from one mode to another will be done by a transition. In the case of adaptive RTES, this change must respect some rules. To manage this adaptability mechanism, we propose to add an association class *Rules* between the *Mode* and *Transition* classes. This class is used to specify how the systems are adapted to its environment changes.

This class *Rules* has two attributes, *OP_RECONF* with the type *NFP-Duration* to indicate the duration of a configuration operation. The second one is *Safe State* with the *Boolean* type, to indicate when the system is in a stable state. This property ensures that adaptation will not affect the proper functioning of the systems.

RTES is mainly related to its runtime environment for that we propose to add a class *Context* to represent the operational environment. *Context* is a passive unit that carries information about the status of a system property. This active class, stereotyped as *ppUnit*, has one attribute, Context with the String type to indicate the context of activating mode.

In real-time embedded systems, a mode can be started by one or more contexts, and a context can launch only one mode. To manage adaptation element, the class *Configuration* is proposed to represent the generalization of adaptation and the change of elements. The *NextMode* method, allow the migration of the system to a new mode that fulfills the system requirements. This active class is stereotyped as an *RtUnit* and have the attributes *extTime*, *memoryUsage* and *DataSize*.

Switching from one mode to another mode will be through a configuration. A *condition* of use is attached to each *configuration* of the system, since a system can have several configurations in one mode, the condition makes it possible to decide the choice of the configuration in a specific context. The conditions link the configurations and each variability of the environment. Every attribute of the *Condition* class captures an element of the environment. The attributes of the *condition* class can be: *battery* which indicates the remaining percentage of the battery, *Memory* which represents the internal state of a memory in percentage. A configuration is composed of one or many schedulable tasks that will be allocated to one or many execution processors.

schedulable Task is a class for modeling the software resource managed by the OS, a task should have a software synchronization resource used to notify events. It is annotated with *schedulableResources* and *NotificationResources*. It has only one attribute, *DeadlineElements*. The class *Execution host*, is stereotyped *SaExecuteHoste*, annotate processing unit. To partitioning tasks in the different scheduler, we use a scheduler annotated *GaExecHost*. The different components of an RTES should be communicated; for that, we add a class communication resources annotated with *Hw- CommunicationResource* which may be a bus or Direct Memory Access.

5 CASE STUDY

To better explain our proposal, we use a UGV (Unmanned Ground Vehicle) (Etienne, 2009) system as an entire running application in this paper. We have chosen this system which is particularly relevant for the dynamic nature of its operational environment and its ability to operate in several modes of operation.

The software architecture of the UGV consists of several subsystems. The communicate between then it will be with sending messages via their communication ports. Figure 3 shows the software architecture of the system.

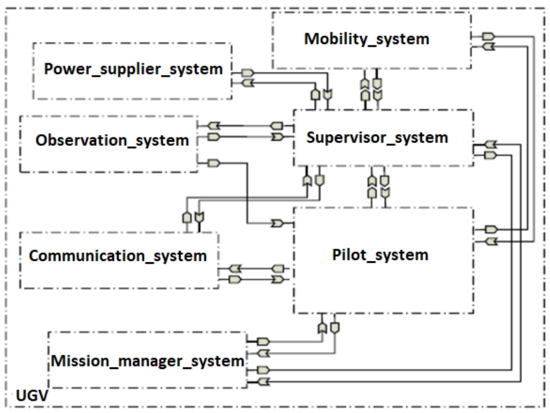

Figure 3: Software architecture UGV (Etienne, 2009).

We are interested in our study, within the robot UGV, *Pilot_System*0 on the control system that operates in two operational modes. An automatic mode, where the subsystem calculates the steering commands from its current position, its environment as perceived by its sensors and a path that has been preprogrammed. A *manual mode*, where the steering system receives steering commands sent by an operator (Figure 4).

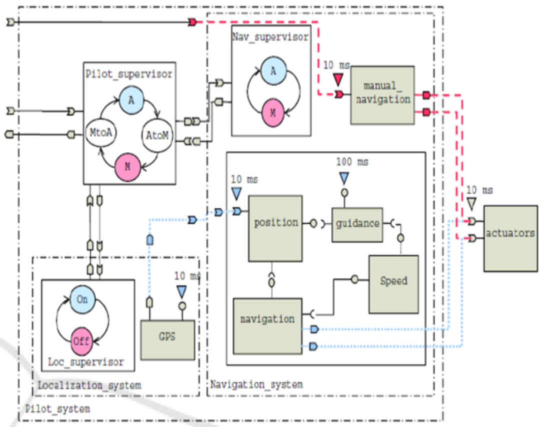

Figure 4: Architecture of the Pilot_System (Etienne, 2009).

As indicated in figure 5, The UGV system is modeled by two modes. The first mode was represented by *M* and indicate the *manual mode* and the second mode is *A* represented the *automatic mode* of the vehicle. The transition from the mode *M* to the mode *A* is represented respectively by the two transitions *MtoA* and *AtoM*. Since the system operates in the *manual mode*, it can switch to *automatic mode* when the communication system is disconnected or by user request. However, if the location system is out of order, this passage is not allowed. Besides, the system switches from the mode *A* to the mode *M* if the location system is not enabled or from user requirement. The availability of the communication system conditions this transition.

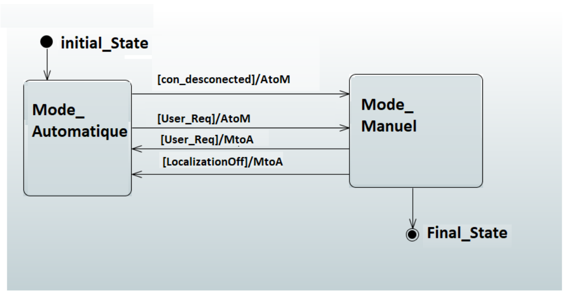

Figure 5: Mode Behavior/Intra Mode.

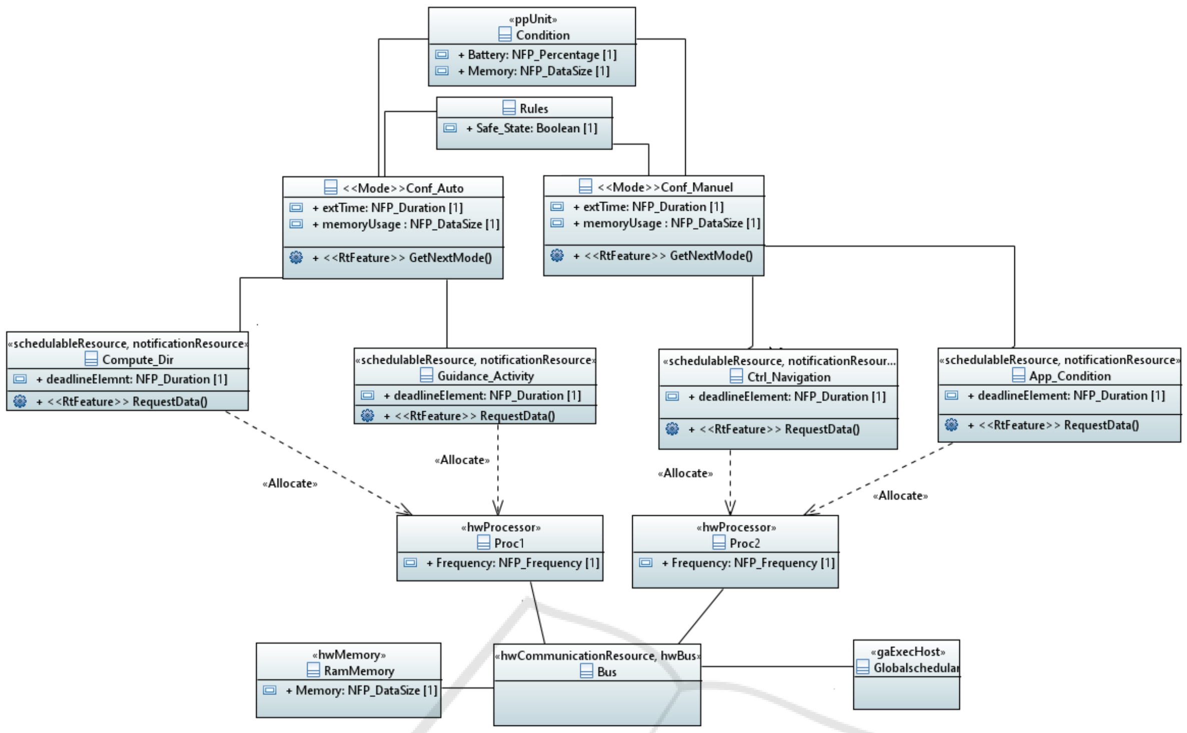
«ppUnit» Condition
+ Battery: NFP_Percentage [1]
+ Memory: NFP_DataSize [1]
Rules
+ Safe_State: Boolean [1]
<<Mode>>Conf_Auto
+ extTime: NFP_Duration [1]
+ memoryUsage : NFP_DataSize [1]
+ <<RtFeature>> GetNextMode()
<<Mode>>Conf_Manuel
+ extTime: NFP_Duration [1]
+ memoryUsage : NFP_DataSize [1]
+ <<RtFeature>> GetNextMode()
«schedulableResource, notificationResource» Compute_Dir
+ deadlineElemnt: NFP_Duration [1]
+ <<RtFeature>> RequestData()
«schedulableResource, notificationResource» Guidance_Activity
+ deadlineElement: NFP_Duration [1]
+ <<RtFeature>> RequestData()
«schedulableResource, notificationResour... Ctrl_Navigation
+ deadlineElement: NFP_Duration [1]
+ <<RtFeature>> RequestData()
«schedulableResource, notificationResource» App_Condition
+ deadlineElement: NFP_Duration [1]
+ <<RtFeature>> RequestData()
«Allocate»
«Allocate»
«Allocate»
«Allocate»
«hwProcessor» Proc1
+ Frequency: NFP_Frequency [1]
«hwProcessor» Proc2
+ Frequency: NFP_Frequency [1]
«hwMemory» RamMemory
+ Memory: NFP_DataSize [1]
«hwCommunicationResource, hwBus» Bus
«gaExecHost» Globalschedular

Figure 6: UGV Class Diagram.

In Figure 7, the *automatic mode* execution scenario is triggered by the *Auto_pilot_ activity* event retrieving the current vehicle position and invoking the schedulable task *Compte_ dur*. This task calculates the commands to be sent to the robot actuators, starting from the current position and the last calculated speed value, the time of this task is 15 ms.

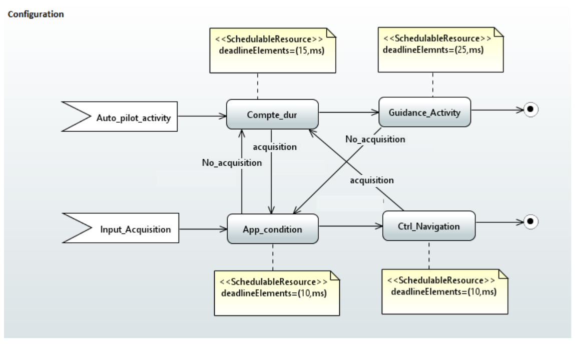
Configuration
<<SchedulableResource>> deadlineElements=(15,ms)
<<SchedulableResource>> deadlineElemnts=(25,ms)
Auto_pilot_activity
Compte_dur
Guidance_Activity
acquisition
No_acquisition
No_acquisition
acquisition
Input_Acquisition
App_condition
Ctrl_Navigation
<<SchedulableResource>> deadlineElements=(10,ms)
<<SchedulableResource>> deadlineElements=(10,ms)

Figure 7: Workload Model UGV system.

Then, the *Guidance_Activity* task retrieves the last ten saved positions to compute the current robot speed, the time of this task is 25 ms. A following link links operations in this mode. On the other hand, the *manual mode* is activated by the input *Input_Acquisition* which corresponds to an order issued by the end user. Subsequently, the *App_ condition* task calculates the guidance commands to be applied, the time of this task is 10 ms. Finally, the *Ctrl_Navigation* action stores the last ten positions of the vehicle in 4ms.

For the automatic mode, we can change from any task to the first task of manual mode if a notification of acquisition found. The same for a switch from manual to automatic mode, if notification das not exist, we can change from any task of the manual mode to the first task of the automatic mode.

As indicated in the figure 6, we have two configurations, *Automatic* and *Manuel* configuration; they were presented respectively in the class diagram with *Con_Auto* and *Con_Manuel*. For the *Con_Auto*, we have two schedulable tasks: *Compte_ Dur* and *Guidance_ Activity*. For the *Con_Manuel* we have two schedulable tasks too, *App_Condition* and *Ctrl_Navigation*.

Each configuration characterises with her *ex_Time* type (*NFP_Duration*), and with her memory usage. For the passage from one mode to another, a condition about the percentage of battery should be validated, in the same time an affirmation about the state of the system type *Boolean* should be verified (It's safe to change from a configuration to another). A schedulable task should be allocated to an execution processor, in our study, we proposed two processor *proc1* and *proc2*, each processor for each configuration. Each execution processor needs same resource to run such as *hwMemory* and a *communication Bus* to relate resources.

6 CONCLUSIONS

Throughout this paper, we have proposed a new design pattern providing support for adaptive modeling and time constraint for RTES. We used UML annotated with the MARTE profile to design our pattern in a static view. The advantage of our approach is the capability to model adaptive properties which will be extracted, to serve during the scheduling step. Our proposal has already been performed on the papyrus tool, which is an editor of MARTE-based modeling, and validated through a case study.

The future task we have assigned to ourselves is to define empirical and comparative studies to provide quality indicators to reduce the development risks of Adaptive RTES and to measure the benefits of our proposal.

REFERENCES

Abuseta, Y. and Swesi, K. (2015). Design patterns for self adaptive systems engineering. *arXiv preprint arXiv:1508.01330.*

Buschmann, F., Henney, K., and Schimdt, D. (2007). Pattern-oriented software architecture: on patterns and pattern language. volume 5. John wiley & sons.

Etienne, B. (2009). Configuration et reconfiguration des systèmes temps-reél répartis embarqués critiques et adaptatifs. In *Télécom ParisTech.*

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

Hamerski, J. C., Abich, G., Reis, R., Ost, L., and Amory, A. (2018). A design patterns-based middleware for multiprocessor systems-on-chip. In *2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6. IEEE.

Magdich, A., Kacem, Y. H., Kerboeuf, M., Mahfoudhi, A., and Abid, M. (2018). A design pattern-based approach for automatic choice of semi-partitioned and global scheduling algorithms. volume 97, pages 83–98. Elsevier.

Magdich, A., Kacem, Y. H., Mahfoudhi, A., and Kerboeuf, M. (2014). A uml/marte-based design pattern for semi-partitioned scheduling analysis. In *2014 IEEE 23rd International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprise (WETICE)*, pages 300–305. IEEE.

Magdich, A., Kacem, Y. H., Mahfoudhi, A., Kerboeuf, M., and Abid, M. (2015). Real-time design patterns: Architectural designs for automatic semi-partitioned and global scheduling. In *International Conference on Enterprise, Business-Process and Information Systems Modeling*, pages 447–460. Springer.

Naija, M. and Ahmed, S. B. (2016a). Extending uml/marte-sam for integrating adaptation mechanisms in scheduling view. In *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, pages 84–90. SCITEPRESS-Science and Technology Publications, Lda.

Naija, M. and Ahmed, S. B. (2016b). A new marte extension to address adaptation mechanisms in scheduling view. In *International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 27–43. Springer.

Naija, M., Ahmed, S. B., and Bruel, J.-M. (2015). New schedulability analysis for real-time systems based on mde and petri nets model at early design stages. In *Software Technologies (ICSOFT), 2015 10th International Joint Conference on*, volume 1, pages 1–9. IEEE.

Naija, M., Bruel, J.-M., and Ahmed, S. B. (2016). Towards a marte extension to address adaptation mechanisms. In *High Assurance Systems Engineering (HASE), 2016 IEEE 17th International Symposium on*, pages 240–243. IEEE.

OMG (2008). Object management group. a uml profile for marte: Modeling and analysis of real-time embedded systems. In *Beta2, Object Management Group.*

Ramirez, A. J. and Cheng, B. H. (2010). Design patterns for developing dynamically adaptive systems. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 49–58. ACM.

Rohnert, F. B. R. M. H. and Stal, P. S. M. (1996). Pattern-oriented software architecture: A system of patterns. page 13.

Said, M. B., Kacem, Y. H., Kerboeuf, M., Amor, N. B., and Abid, M. (2014). Design patterns for self-adaptive rte systems specification. *International Journal of Reconfigurable Computing*, 2014:8.

Schmidt, D. C. (2006). Model-driven engineering. volume 39, page 25. Citeseer.

Supriana, I., Surendro, K., et al. (2018). Self-adaptive software modeling based on contextual requirements. *Telkomnika*, 16(3).