**American Journal of Embedded Systems and Applications**
2017; 5(6): 54-59
http://www.sciencepublishinggroup.com/j/ajesa
doi: 10.11648/j.ajesa.20170506.13
ISSN: 2376-6069 (Print); ISSN: 2376-6085 (Online)

# Towards a Meta-Model for Real-Time Embedded Systems

**Soukaina Moujtahid, Abdessamad Belangour, Abdelaziz Marzak**

Faculty of Science Ben M'Sik, University Hassan II, Casablanca, Morocco

**Email address:**
soukainamoujtahid@gmail.com (S. Moujtahid)

**To cite this article:**
Soukaina Moujtahid, Abdessamad Belangour, Abdelaziz Marzak. Towards a Meta-Model for Real-Time Embedded Systems. *American Journal of Embedded Systems and Applications*. Vol. 5, No. 6, 2017, pp. 54-59. doi: 10.11648/j.ajesa.20170506.13

**Received**: December 5, 2017; **Accepted**: December 18, 2017; **Published**: January 11, 2018

**Abstract:** Embedded real-time systems are combinations of hardware and software fully integrated into the systems they control. Due to the continuous technological evolution in the hardware and software and the diversity of the targeted areas of application, these systems have become omnipresent in our professional and personal lives. Thus, various approaches based on model driven engineering (MDE) have been proposed in order to control the inefficiency of the methods of their current design. Each of these approaches has its own meta-model and its corresponding UML profile, specialized or adapted to a particular category of these systems. Indeed, in this paper we will propose a generic meta-model, taking advantage of a large number of these meta-models, which can be adapted to the majority of embedded real-time systems.

**Keywords:** Real-Time Embedded System, MDE, Meta-Model

## 1. Introduction

Embedded real-time systems are now omnipresent in various fields. Their intelligence makes them more and more indispensable [17]. They are, however, characterized by complexity arising from their specific characteristics and the high industrial constraints to which they are subjected. To master this complexity, various software abstractions have been implemented [18]. The abstraction offered by model driven engineering (MDE) provides an adequate framework for mastering this complexity. It is a new discipline of software engineering that advocates the massive use of models throughout the software development process. Thus, various approaches based on the MDE have been created by proposing their own process of development of the embedded real-time systems [6]. Our research work lies within of this problem. We aim to propose a new model-based development process for embedded real-time systems. To this end, we have begun by characterizing these systems. We have tried to collect the characteristics of these systems which distinguish them from other computer systems. Then, we studied the different development processes based on models proposed for these systems while carrying out a comparative and detailed analysis of the most well-known meta-models. The aim object of this article is a generic meta-model taking advantage of these different meta-models. Thus, this paper is structured in four parts: The first gives a general view on embedded real-time systems and more precisely on their characteristics. The second part presents the integration of the MDE in the process of development of embedded real-time systems via a panorama of the most known meta-models and on which we were based to create a generic meta-model. This last one which will be the subject of the third section and finally the fourth section concludes with a synthesis of the paper.

## 2. Characterization of Embedded Real-Time Systems

A real-time system is a computer system to control the behavior of a physical process to which it is connected [1]. To do this, a real-time system consists essentially of a software application controlling the process, a hardware and/or software execution support executing this application, sensors supplying the input data of this application, and actuators executing the orders produced by this application. The application, the execution platforms, the sensors and the actuators are then embedded. They are buried in the process to which they are connected [2]. Embedded systems are combinations of hardware and software [1] completely

integrated in the system they control. The complexity of the development of embedded systems stems from the specific characteristics of embedded systems that distinguish them from purely software systems (measurement and control of the physical world, execution on a physical platform limited in resources, autonomy, reliability, reactivity, Etc.), and on the other hand, the high industrial constraints to which these systems are subjected: development and manufacturing costs and delays, multidisciplinary teams, certification and documentation etc. Various methods and languages were proposed in order to master this complexity by emphasizing the modeling of the application and the platform constituting the embedded system [4]. Embedded real-time systems differ from conventional software systems by a set of features. The characterization of these systems will allow us to understand their specificities in order to propose a development approach that is specific to them. However, we can classify these specificities into two categories: characteristics specific to embedded systems and others to real-time systems. Embedded real-time systems are characterized by a memory footprint, energy consumption, weight and volume, autonomy, mobility, communication, security constraints, cost of products in relation to the target sector and other characteristics which will be seen in detail [6] [3]. On the other hand, they have a common characteristic that resides in the existence of temporal constraints to be taken into account. These constraints can take various forms such as deadlines, time intervals, duration of validity, etc. And apply to various objects [1] [2] [4]. The data have a limited lifetime and become obsolete after a certain time, the events appear at special moments and must be taken into account after known delays and the treatments often have moments of beginning, end and fixed execution times. Therefore, these systems work in real-time to manage information and to deduce actions in a controlled time [4]. This means that the accuracy of the results of these systems depends not only on the functional aspects but also on the time in which these results were obtained [3]. Embedded real-time systems are present in several fields [14] such as healthcare, the automotive industry, telecommunications, aeronautics, commerce, household appliances, etc. in addition to their traditional fields such as military or Spatial [1] [2] [3] [11]. Thus, for example, an on-board computer is a computer integrated in a vehicle that collects, in real time, information about the condition of the vehicle such as fuel level, oil level, door opening, seat belt buckle, speed control, etc. It is certainly unnecessary if this information is available only after a significant period of time [5]. So the on-board computer is an embedded real-time system.

# 3. Modeling Embedded Real-Time Systems with the MDE

The development of quality embedded real-time systems at controlled costs represents a very important technological and industrial challenge. In some industrial sectors, such as transport, aeronautics, or telecommunications, the maintenance of these quality / cost objectives implies the use of formal validation and verification tools and techniques [4]. This requires the use of a model-oriented approach. One of the characteristics of this approach is the use of executable models that contain information about the different aspects of the system: functional requirements, static architecture, no-functional requirements, etc. For the use of such models to be relevant, the formalism used must be sufficiently expressive, and its use must be supported, not only by tools for code generation, but also by validation and formal verification tools at the level of the model [10]. The OMG has defined the MDA approach (Architecture Directed by the Models). The key objective of this approach is to develop business models independent of the execution platforms (PIMs) and to transform them into models specific to a given execution platform (PSM) such as Java / J2EE, ASP.Net, etc. This was to preserve information systems against technological changes and increase productivity. The transition from PIM to PSM involves a platform description model (PDM) and model transformation mechanisms [5]. MDE is a discipline that is a continuation of post-object technologies, such as UML, design patterns and the MDA [6]. It generalizes the idea of MDA where the goal is not limited to the PIM-PSM transformation but to the use of the models in all the activities of the software production. Each model conforms to a meta-model expressed by the meta-model MOF (Meta Object Facility) [10]. As part of a model-driven engineering we used the following meta-models to create a generic meta-model:

SPT (Schedulability, Performance and Time) is a profile adopted by OMG [17] based on version 1.3 of UML [10], taking into account the specificities of real time while retaining the benefits of the object-oriented approach, and provides a framework for temporal modeling [17] of system planning processes that can be used during the making-decision process [12]. SPT's goal was to fill the gaps in AML (Abstract Modeling Level) 1.4 for designers and developers of real-time applications by allowing annotation of model elements by quantitative information about time [17]. This information is then used for performance analyzes based on models, scheduling or verification of compliance with real-time constraints [15]. SPT considers only a metric time that implicitly refers to physical time. SPT introduces the concepts of instant (moment) and duration (duration), as well as those of time-related events and stimuli. SPT also models timing mechanisms and associated services (start, stop, suspend, resume). All this via stereotypes to be applied in UML modeling elements to specify the time values "RTtime", "RTclock", "RTtimer", "RTtimeout", "RTdelay", "RTintervale" etc. [17]. The general structure of SPT as shown in the figure below consists of various sub-profiles [17].

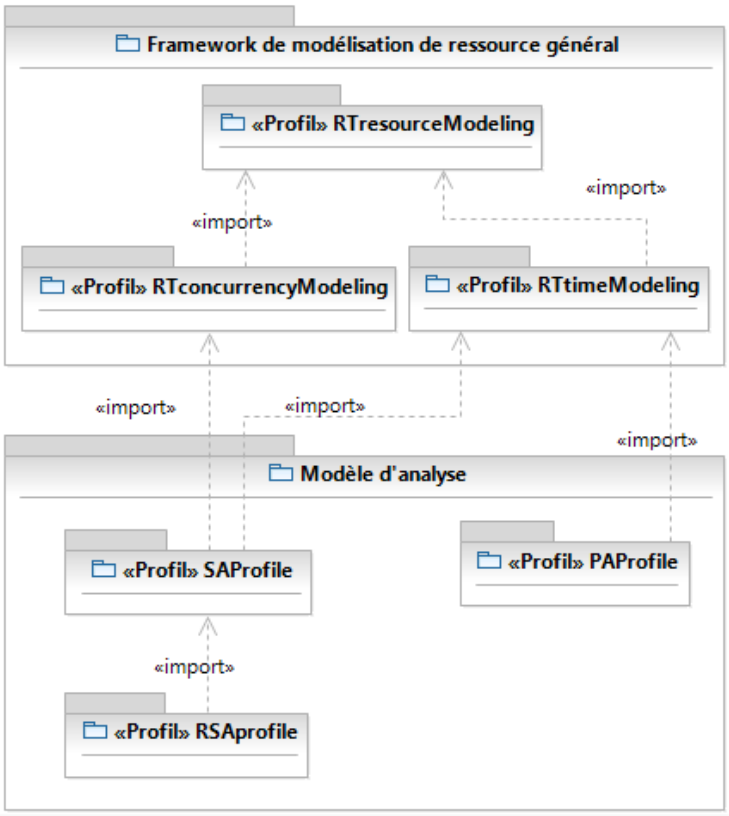


***Figure 1.*** *SPT meta-model.*

However, SPT has power gaps and flexibility problems. Hence SPT has been replaced by MARTE [15].

Modeling and Analysis of Embedded Real-Time Systems (MARTE) is a UML profile [14] which follows the SPT profile [10]. MARTE enriches UML with new concepts in order to analyze and model software as well as the hardware of embedded real time systems [6]. The architecture of the profile, as shown in the figure below, consists of the packages: foundations, design model, analysis model and appendices [9].

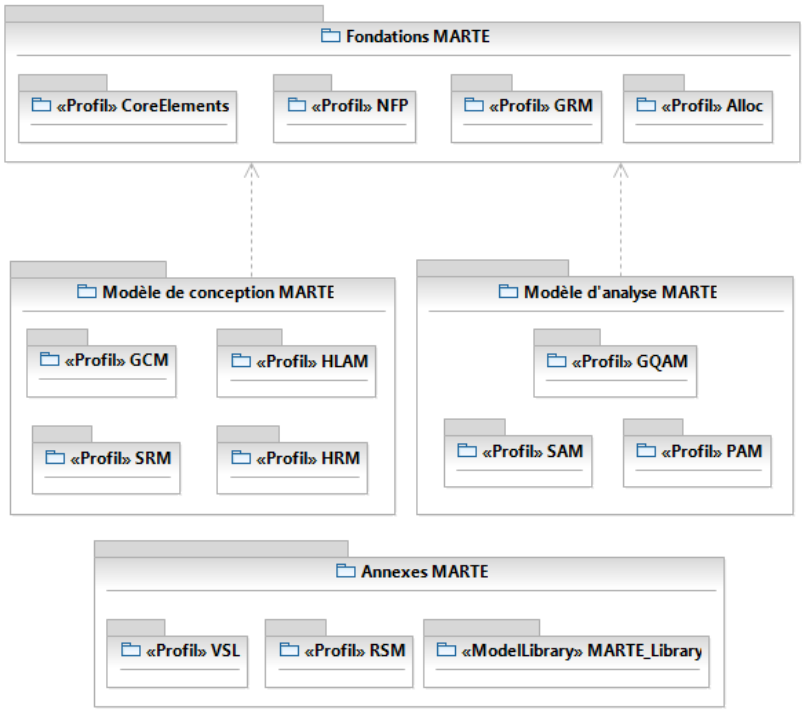


***Figure 2.*** *Marte meta-model.*

The basic concepts in the MARTE meta-model are defined in the Core Elements sub-package (such as Model Element, Classifier, Instance, etc.) [9] which is itself divided into two packages: Foundations and Causality. The first introduces the concepts of Model Elements, Classifer, and Instance. The second deals with behavioral concepts (common Beravior) in which events, executions and execution contexts are introduced. Because of these multiple concepts, MARTE presents a great difficulty of use [15].

MASTE (Modeling and Analysis Suite for Real Time Applications) defines a meta-model to describe the temporal behavior of real-time systems to be analyzed by scheduling analysis techniques. MAST also provides a set of open-source tools for performing scheduling analysis or other time analysis to determine whether the system will be able to meet its time requirements. Tools are also provided to assist the designer in assigning scheduling parameters. By having an explicit model of the system and automatic analysis tools, it is also possible to carry out the design of space exploration. A discrete event simulator is also provided to obtain statistical information on the performance of the modeled system [13].

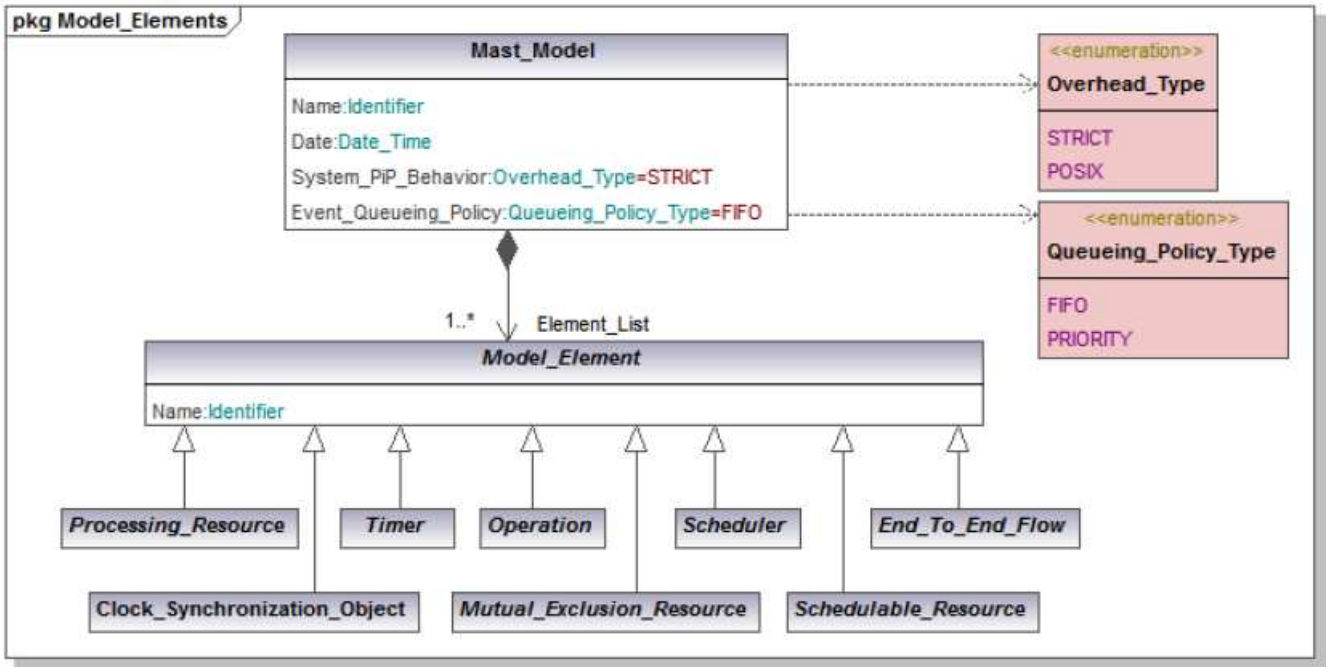


***Figure 3.*** *Maste meta-model.*

For general design and basic concepts, each of these meta-models uses its own concepts based on the UML meta-model. MASTE represents only the time part of the embedded real-time systems, on the other hand MARTE and SPT allows in addition the presentation of the hardware part but with various notions which makes their implementation difficult.

## 4. Proposition of a Generic Meta-Model for Embedded Real-Time Systems

We have noted in the above that there is not a complete meta-model covering all aspects related to embedded real-time systems and easy to be interpreted. However the coupling between them is possible and can lead to better results. This coupling is possible since most of the profiles are focusing on the process paradigm. What we need is a meta-model, in which we must define rules for the automatic transition from one meta-model to another. This led us to work on a generic meta-model representing an integration of these meta-models: SPT, MARTE and MAST. Below we will give examples of parts of this meta-model. The key elements of this meta-model are resources, services provided by these resources and operations given by them. This resource can be hardware or software. A software resource can be a resource of calculation, communication, backup, time, synchronization etc. The captures below give a general view of the most important parts of this meta-model:

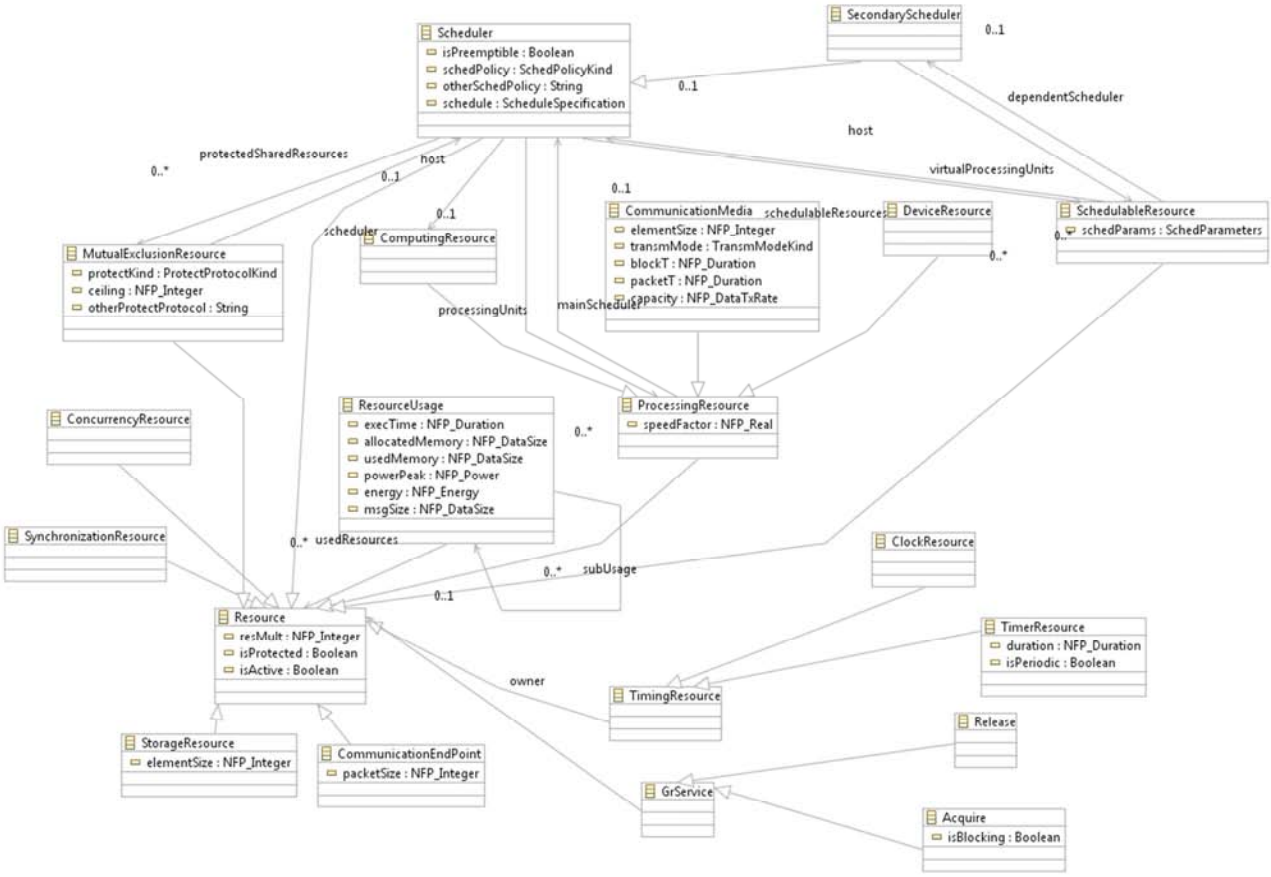


***Figure 4.*** *Resource model.*

This model provides an abstract definition of the resource and its different uses. The type of use of a resource depends on the type of this resource. For example, each dynamic object creation consumes memory and CPU time. We can precisely specify the amount of resource used, by using the class Resource Usage. This class includes a number of optional parameters, including the amount of memory required by a service request (assigned Memory).

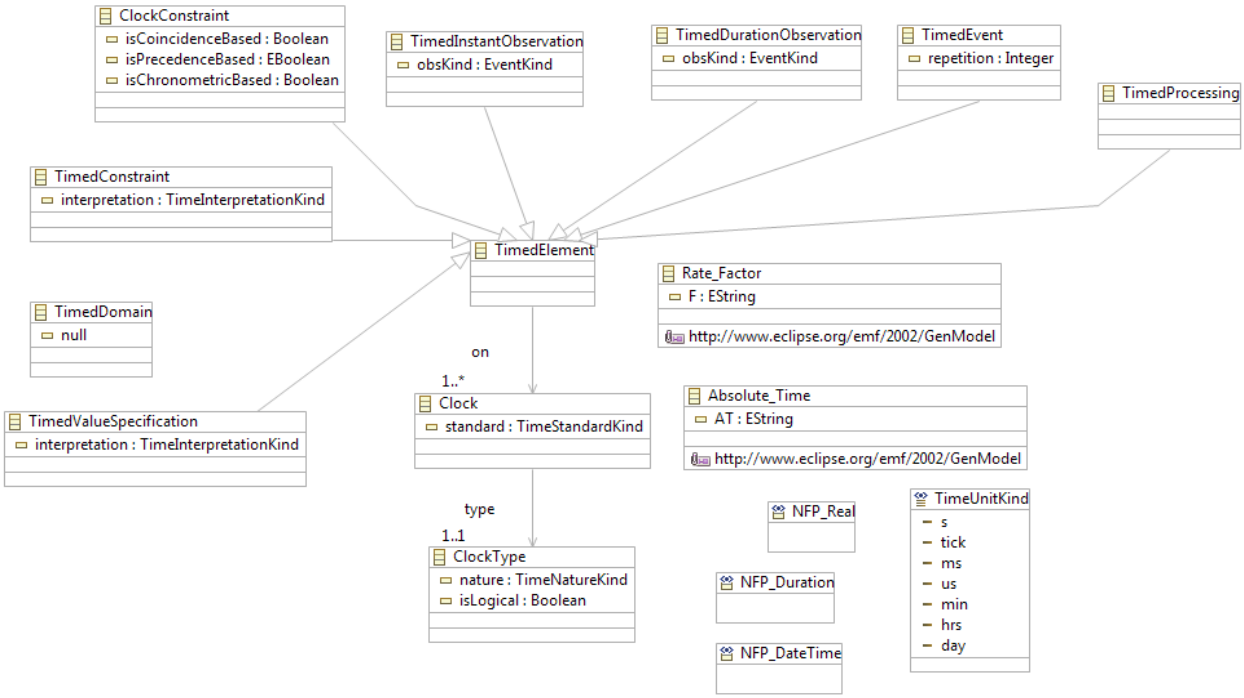


***Figure 5.*** *Time model.*

Time plays a crucial role in real-time and embedded software applications and needs to be properly accounted for in many kinds of engineering analyses. This model describes the timing behaviour of real-time systems designed to be analyzable via schedulability analysis techniques. It allows time and its progress to be associated with model elements that represent dynamic behavior, such as operation invocations, messages, state machine transitions, actions and activities, and so forth.

# 5. Conclusion

We have tried to present in the different sections of this article the contribution of MDE in the modeling process of embedded real-time systems. This is done through its specialized meta-models adapted to its systems. We can find that each of the profiles has these own advantages, although each of them applies to a particular category of embedded real-time systems. It would be more benefit to take advantage of these specificities to have a meta-model suitable for the majority of embedded real-time systems. It is in this context that was presented in the last section the generic meta-model proposed.

# References

[1] Embedded Systems Design for High-Speed Data Acquisition and Control. Springer International Publishing Switzerland (2015)

[2] Peter Hintenaus: Engineering Embedded Systems. Physics, Programs, Circuits. Springer International Publishing (2015)

[3] Dynamic Memory Management for Embedded Systems. Springer International Publishing Switzerland (2015)

[4] Nicolas Hili: Une méthode pour le développement collaboratif de systèmes embarqués (2014)

[5] Model-Driven Engineering of information systems. Principles, techniques and practice. Apple Academic Press, Inc. (2015)

[6] Wiley-ISTE: Model-Driven Engineering for Distributed Real-Time Systems MARTE Modeling, Model Transformations and their Usages (2010)

[7] Pascal Roques. Modélisation de systèmes complexes avec SysML. Broché (2013)

[8] Nicolas Belloir, Jean-Michel Bruel, Raphael Faudou: Modélisation des exigences en UML/SysML. (2014)

[9] Mohamed-Lamine Boukhanoufa: Approche basée sur les modéles pour la conception des systémes dynamiquement reconfigurable: de MARTE vers RecoMARTE. (2013)

[10] Elsevier Inc. Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE. Developing Cyber-Physical Systems". Bran Selic et Sébastien Gérard (2014)

[11] Jon Holt et Simon Perry. Sys ML for Systems Engineering. A model-based approach. The Institution of Engineering and Technology (2013)

[12] Samba Diaw et Rédouane Lbath et Bernard Coulette. “SPEM4MDE: un métamodèle basé sur SPEM 2 pour la spécification des procédés MDE”. Majec STIC 2009. Avignon, France. 18 novembre (2009)

[13] César Cuevas Cuesta, José María Drake, Michael González Harbour, José Javier Gutiérrez, Patricia López Martínez, Julio Luis Medina et José Carlos Palencia. Modeling and Analysis Suite for Real Time Applications. Universidad de Cantabria, SPAIN (2010)

[14] Alberto Sangiovanni-Vincentelli, Haibo Zeng, Marco Di Natale et Peter Marwedel. Embedded Systems Development. From Functional Models to Implementations. Springer Science+Business Media New York (2014)

[15] Muhammad Waqar Aziz, Radziah Mohamad et Dayang N. A. Jawawi. Critical evaluation of two UML profiles for Distributed Embedded Real-Time Systems Design. International Journal of Software Engineering and Its Applications Vol. 7, No. 3 (2013)

[16] Mohd Zulkifli Bin Mohd Zaki et Dayang Norhayati Binti Abang Jawawi: A Review on UML-RT and UML-SPT for Embedded Real-Time Component-Based Development (2013)

[17] Jiacun Wang: Real-Time Embedded Systems. 7 John Wiley & Sons, Inc. (2017)

[18] Embedded and Real-Time Operating Systems. Springer International Publishing (2017)