A MARTE extension for global scheduling analysis of multiprocessor systems

Amina Magdich, Yessine Hadj Kacem, Adel Mahfoudhi, and Mohamed Abid
CES Laboratory, ENIS Soukra km 3,5
B.P.: 1173-3000 Sfax TUNISIA
Email: amina.magdich, yessine.hadjkacem, adel.mahfoudhi, mohamed.abid@ceslab.org

***Abstract*—Real-Time Systems are subject to Soft/Hard temporal constraints. Besides, a tasks scheduling step is required to meet the maximum of deadlines, for which there are different scheduling algorithms to do. However, nowadays real-time systems represent a serious issue for the worldwide industry due to their growing complexity. Indeed, since they are more susceptible to failures and deficiencies of development, it is crucial to rely on high level development methods. In this context, some researchers have proposed scheduling analysis within the new profile Modeling and Analysis of Real-Time and Embedded systems (MARTE), which supports both monoprocessor and multiprocessor scheduling algorithms. While supported multiprocessor scheduling algorithms are part of the partitioned approach, global approach algorithms have not been backed by MARTE yet. In the present paper, we seek to improve the meta-models of MARTE stereotypes to establish scheduling analysis for the global approach.**

***Keywords*-multiprocessor scheduling analysis; global approach; MDE; MARTE;**

I. Introduction

Real-time systems are facing a set of development difficulties, among which the tasks scheduling can be stated. Thus, developers are encountering the challenge of computing resource recovery, preemption time, the possibility of task's migration, etc. This problem should be solved by schedulability theory [2][5]. Nonetheless, a real-time system has to meet the temporal determinism; i.e., it has to respect the time constraints to which it is exposed. What is worthwhile to note is that a schedulability analysis is necessary to verify the behavior of the studied system. Furthermore, user requirements evolve by the time. Since complex real-time applications impose a set of principal constraints, the use of simple monoprocessor architecture does not always meet their needs and the constraints that are required by designers. Additionally, since it would be more useful to rely on parallel/multiprocessor architecture, the complexity of such systems increases. This necessitates high level development techniques to overcome this challenge. In the same vein, the Model Driven Engineering (MDE) [18] is a way to beat the shortcomings of complex electronic systems development. In particular, Unified Modeling Language (UML) profiles promotes an adequate solution to support the whole life cycle co-design of complex systems with their real time constraints and performance issues. In this context, the present paper focuses on the use of MARTE profile [7] within the schedulability analysis context [19]. Two major scheduling approaches are available in the literature: The partitioned and the global approaches. MARTE supports only algorithms for the partitioned approach; it does not allow task's migration. Thus, we propose in this paper the main changes to be made on MARTE profile for supporting global scheduling approach. These amendments affect mainly the stereotypes of MARTE/SAM (Scheduling Analysis Modeling) since it is the sub-profile intended to model the schedulability analysis. Our contribution is used to change the meta-models of existing stereotypes such as gaAcqStep, gaRelStep, saStep, gaStep and gaScenario. We try to modify in the structure of the corresponding existing attributes, with no need to add new ones, by referring to their multiplicities which makes easy the adoption of the proposed extension in revisions of MARTE.

This paper is structured as follows. Section 2 emphasizes the various related works. Section 3 defines the recent MDE paradigm and its MARTE profile. Section 4 highlights the importance of using scheduling analysis theory in the context of MARTE annotations. Section 5 specifies the main contribution of this paper. It sets up the various changes attributed to MARTE/SAM meta models. To further explain our contribution, in section 6, we rely on a pedagogic case study. Finally, conclusions and future works are illustrated in section 7.

II. Related Works

Some works are available in the literature reporting the use of MARTE within the schedulability analysis context. For example, in [21], the authors highlighted the importance of using the UML/MARTE profile compared to Schedulability, Performance and Time (SPT) [6]. They focused on the exploitation of MARTE within the context of schedulability analysis. ACCORD/UML [4][20] methodology founded on this profile enabling the schedulability analysis and originally based on the SPT profile is illustrated. Moreover, in [15], the researchers benefited from the use of UML within the schedulability analysis. They attempted to clarify the concepts needed to build an UML model annotated with MARTE stereotypes. The constructed model should cover all the data required to be submitted to the chosen analysis tool. In this work, the authors have chosen the MAST tool. Indeed, other studies have transformed MARTE models to

scheduling analysis tool models. In [9], the authors have benefited from the profile MARTE/SAM to model schedulability analysis of RTES (Real Time Embedded Systems). By the way, they suggested an automatic mapping from UML activity diagram to PTPN [11] for establishing a formal verification. Although the proposed methodology is intended for multiprocessor systems, it is based on the scheduling partitioned approach (using the RM algorithm) that does not allow task's migration. In [10], the authors presented a view based on UML profile for schedulability analysis and used an automatic conversion to SymTA/S scheduling analysis tool. Besides, they have proposed an extension for the papyrus palette. In that way, not to be expected to know the details of MARTE profile, developers can quickly create scheduling analysis views. Moreover, in [8], although the authors have addressed the problem of modeling and scheduling analysis of real-time systems, they didn't take advantage from MARTE. In this context, they have reused in [16] the same flow with the integration of MARTE profile to deal with the same problems. In fact, this article uses a model-driven approach to design Ada applications. It sets a tool that transforms the initial models to schedulability analysis models. The authors took advantage from MARTE stereotypes to annotate models. The schedulability analysis tool used in this paper is MAST. A different orientation was reported in [1]; the researchers have adapted two modeling languages MARTE and EAST-ADL2 in order to design and analyze automotive systems at an early design stage. They employed the MAST tool to decide about the schedulability of the studied system. Furthermore, other researchers have taken advantages from the use of MARTE in the framework of schedulability analysis, but they did not address all the scheduling algorithms types, such as in [3] (the scheduling algorithm used was RMA). This article explains well the different steps needed to reach the schedulability results. Nevertheless, the proposal of this work does not support monoprocessor systems with periodic or sporadic tasks. It rather supports either distributed or multiprocessor systems. In addition, in [17] the authors have also benefited from the MARTE profile in the context of a high level systems development. Their article was structured around two main contributions; Firstly, the use of MARTE for modeling systems that contain both functional and non functional properties, and secondly, the exploitation of used models to extract and implement temporal information in order to provide scheduling analysis. Actually, the authors have melted four basic models, one of which is to present functional and non-functional concepts that represent an alternative model based on time profile containing clocks properties. With respect to the two other models, they are to represent the various resources and the allocation of functional part. The authors have benefited in particular from the model incorporating the clock's characteristics to check tasks schedulability. Nevertheless, the scheduling algorithm used was DM fixed priority. Though this algorithm can be applied either on monoprocessor or multiprocessor architecture, when applied on multiprocessor architecture, it belongs to the partitioned scheduling approach and does not support task's migration.

All the mentioned works have suggested high level approaches to fight against the increasing complexity of real time applications. They benefited from MARTE profile to fulfill the schedulability analysis at a high level of abstraction, which allows the detection of mistakes early. Although these research studies have focused either on monoprocessor or multiprocessor systems, they have not yet dealt with the scheduling algorithms that support task's migration. It is in this context that we propose an improvement to MARTE meta-models, and especially modifications in the structure of SAM meta-models.

III. The Model Driven Engineering

The MDE is a specific software engineering approach aiming at the definition of the theoretical framework to generate code using successive model's transformations. It allows the overcoming of the growing complexity of electronic systems. Furthermore, it has several benefits, among which we can cite the flow's automation, the capture and the validation of constraints (using OCL (Object Constraint Language)), the independence overlook of the technology developments, better management of complexity, better reuse of models, etc. In the context of the schedulability analysis, MDE is mainly used in the modeling step and the transformation of scheduling analysis models to the models of the chosen scheduling analysis tool. Typically, the MDE uses the UML profile and especially MARTE.

A. MARTE

MARTE which is a contribution to the MDE is a specification of UML adopted by OMG (Object Management Group) aiming to replace the UML SPT profile. It does not only bear the model-driven development and real time systems analysis, but also provides a unified co-design of Software/Hardware parts at a high level of abstraction. This profile which is meant to support the real time embedded systems schedulability analysis offers standard annotations. The aim of theses stereotypes is to bring the model system studied from the schedulability analysis tool model, which promotes independence between the different stages of real time systems development and those of schedulability analysis. Then, any schedulability analysis tool could be exploited. MARTE includes a set of sub-profiles such as GRM (Global Resource Modeling) (SRM (Software Resource Modeling), HRM (Hardware Resource Modeling)), GQAM (Generic Quantitative Analysis Modeling) (SAM, PAM (Performance Analysis Modeling)), etc, each of which has its own features.

B. SAM

To avoid errors in the design of complex systems and especially those related to temporal behavior, it is essential to rely on models for schedulability analysis. In this context, MARTE/SAM offers a variety of stereotypes for annotating models with real-time features. This profile has the capacity to predict if all tasks meet their time constraints. Subsequently, it promotes the validation of the system temporal accuracy.

IV. Scheduling analysis Theory and MARTE profile

Given that a set of tasks must be performed on all available computing resources, it is not possible to execute tasks without specifying the run sequence. Moreover, the assignment of tasks on processors must be optimal so that the maximum of deadlines must be satisfied. This problem is solved within the scheduling theory that is built on a set of various algorithms. Scheduling algorithms can be classified as part of monoprocessor or multiprocessor scheduling. Employing the MARTE/SAM profile through a schedulability analysis, it should be noted that several algorithms are supported by this profile. The choice of the scheduling algorithm to be used is done through the attribute «schedPolicy», which is an attribute of «scheduler» stereotype belonging to the MARTE/GRM sub-profile and means the scheduler. Then, the available algorithms are «EarliestDeadlineFirst», «FIFO», «FixedPriority» (DM and RM), «LeastLaxityFirst», «RoundRobin» and «TimeTableDriven».

A. Monoprocessor scheduling and MARTE

Monoprocessor scheduling algorithms concern systems with architecture composed of a single computing resource. They differ in the viewpoint supported types of tasks (periodic or not), types of priorities (static or dynamic), etc. Among the existing monoprocessor scheduling algorithms, MARTE supports for example RM [14] and DM [12]. These two algorithms are also ameliorated to be employed in the context of multiprocessor systems as well, but, used only in the context of partitioned approach. The corresponding annotation is «FixedPriority».

B. Multiprocessor scheduling and MARTE

The multiprocessor scheduling problem has been treated for the first time by Liu [13]. It is a theory for systems with multiprocessor architecture. Indeed, there are three families of multiprocessor scheduling algorithms [5]: the scheduling by partitioning, half partitioned and the global approaches. Obviously, MARTE supports multiprocessor scheduling algorithms like «EarliestDeadlineFirst» and «LeastLaxityFirst». But, these algorithms belong to the partitioned scheduling approach. In that case, multiprocessor scheduling problem should be resolved as a mono processor one. It should be borne in mind that there is no task's migration. Indeed, MARTE does not supply stereotypes to support scheduling algorithms of the global approach. Thereby, in the next sections we propose the solution to solve this limitation.

V. Our contribution

In this section, we specify the different changes made on the various stereotypes used within the schedulability analysis. These amendments relate especially to their attributes. Of course, this will be shown in the structure of stereotypes meta-models.

We have previously mentioned that the attribute «schedPolicy» of the stereotype «scheduler» allows the choice of the selected scheduling algorithm's name. If the scheduling algorithm is unknown, it is imperative to select the option «Undef». However, if the scheduling algorithm is not supported by MARTE, it is crucial to choose the option «Other». It is to be noted that is possible to specify the name of the scheduling algorithm that we wish to use. In the attribute «otherSchedPolicy» (an attribute of the stereotype «scheduler»), we state the name of the desired algorithm. Yet, doing that is not sufficient to support scheduling algorithms for the global approach. It is necessary to study the MARTE stereotypes to overcome the lack of supported scheduling algorithms. We seek to further support global multiprocessor scheduling algorithms like PFair and others. Thus, as noted we tend to alter the structure of some attributes.

The scheduling analysis modeling is performed through the MARTE/SAM profile. In this context, the stereotypes «gaWorkloadEvent», «gaAcqStep», «gaRelStep», «saStep» and «saCommStep» are the most used. The amendments we are trying to bring concern only the attributes of stereotypes «gaAcqStep», «gaRelStep» and «saStep», and more particularly the multiplicities of the corresponding attributes. In the following subsections, we specify the changes made to the attributes of each stereotype.

A. Changes affected to meta-models of specialization stereotypes

A specialization inherits all the attributes of its generalization. So in order to change their structures, it is essential to modify the generalization meta-model. It is worthwhile to note that in the following sub-sections, we intend to identify all the necessary changes that affect the specializations meta-models. However, the corresponding meta-models do not show the modifications affected to inherited attributes.

1) Amendments in the meta-model of the stereotype «gaAcqStep» : The concept of task's migration requires that the schedulable resource can be executed on different computing resources for the same period. Subsequently, in order to support this notion, we need to specify the names of various shared resources that a task can block. It is to be noted that the processors corresponding to the locked resources must be specified. There are the computing

resources on which the task can run for a new period or after a migration. Then, the attributes «acqRes» and «Host» must have a multiplicity of [0..*] (i.e. «acqRes» means the name of the shared resource and «host» specifies the name of the computing resource). Otherwise, in the case of multiprocessor systems with dynamic priority, a task can change its priority, thus, the attribute «priority» must have a multiplicity of [0..*]. Moreover, since a task can be divided into a number of instances, the multiplicity of the attribute «concurRes» which indicates the name of the currently job (task's instance) must be [0..*] (Figure 1 and Figure 2).

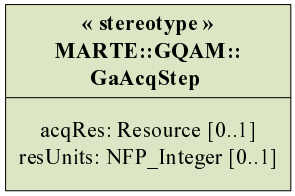

Figure 1. The old meta-model of «gaAcqStep»

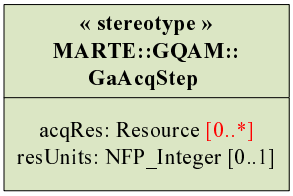

Figure 2. The proposed meta-model of «gaAcqStep»

2) Changes in the attributes of the stereotype «gaRelStep» : Once a task's execution is finished or interrupted, it does not only release the previously-blocked shared resource, but also frees the corresponding processor. Since we treat the case of multiprocessor scheduling with task's migration, a task can allocate different resources and various computing resources (It blocks a different resource after each migration or for each new period). Subsequently, it can just release these different resources when interrupted or terminated. So, the multiplicity of «relRes» and «Host» must be [0..*]. The release of the shared resources is dependent on the end of task's execution, the interruption from a higher priority task or as a self interruption. The reason of the mutual exclusion resource's release is denoted by the attribute «cause». Since the task can unlock the shared resources under different conditions and many times for the same period, the multiplicity of the attribute «cause» must be [0..*]. Moreover, as in the case of the stereotype «gaAcqStep», the attribute «priority» must have a multiplicity [0..*]. Regarding the stereotype «gaAcqStep», its attribute «concurRes» must have a multiplicity of [0..*] (Figure 3 and Figure 4).

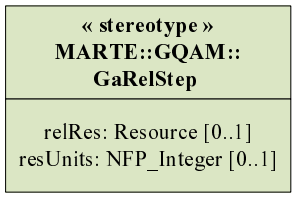

Figure 3. the initial meta-model of «gaRelStep»

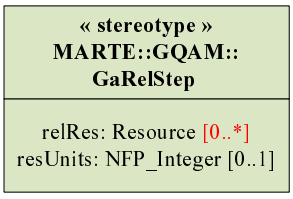

Figure 4. new meta-model of «gaRelStep» after changes

3) Changes in the attributes of the stereotype «saStep» : «saStep» is a stereotype that defines an action and can be used to set the action of performing a task. In the case of a multiprocessor scheduling within task's migration, a task can be interrupted several times during one period. Subsequently, the attribute «preemptT» must have a multiplicity [0..*] (i.e. «preemptT» represents the period of time during which the task is interrupted). Similarly for the attribute «readyT» which refers to the time at which a task can start its execution. This attribute must have a multiplicity of [0..*], and of course, since we deal with the scheduling with task's migration, a task can be executed on different computing resources. Therefore, identically to the case of the stereotypes «gaAcqStep» and «gaRelStep», the attribute «Host» must have a multiplicity [0..*]. In addition, as noted before, the complexity of the attribute «concurRes» must be [0..*]. Moreover, the context of switching causes the change of some attributes such as «deadline», «priority», etc. Then the attribute «deadline» should have a multiplicity of [0..*] (Figure 5 and Figure 6).

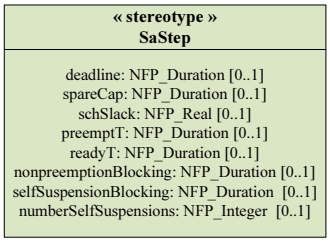

Figure 5. the old attributes structure of the stereotype «saStep»

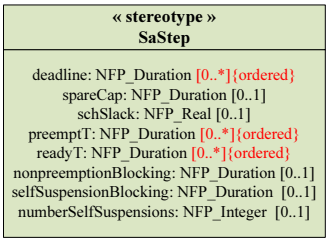

Figure 6. the new meta-model structure of the annotation «saStep»

B. Changes affected to meta-models of generalization stereotypes

In the following subsections, we illustrate the amendments affected to the attributes of generalization meta-models.

1) Modifications in the attributes of the stereotype «gaStep» : We have noticed that some attributes such as «Host», «priority», «concurRes» and «Cause» are common among several stereotypes such as «gaAcqStep», «gaRelStep» and «saStep». To change the structure of these attributes, we do not need to modify the corresponding meta-models, but rather the meta-model of the generalized stereotype. Since «gaAcqStep», «gaRelStep» and «saStep» are specializations of «gaStep», they inherit all its attributes. Eventually, it is just necessary to change the attributes of «gaStep», so that these changes take effect in the meta-model of each specialization (Figure 7 and Figure 8).

2) Changes in the attributes of the stereotype «gaScenario» : As noted above, the attribute «cause» must have a multiplicity of [0..*]. This attribute is common to several stereotypes such as «gaAcqStep», «gaRelStep» and «saStep». Thus, instead of changing its structure in

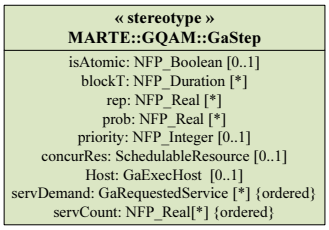

Figure 7. The old meta-model of «gaStep» stereotype

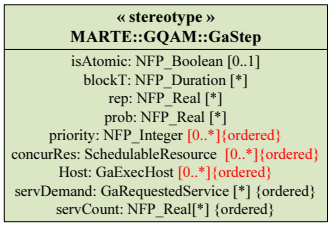

Figure 8. The new meta-model of «gaStep» with the proposed modifications

the mentioned stereotypes; it is sufficient to modify in the meta-model of the stereotype «gaScenario». Indeed, «gaScenario» represents a generalization of «gaStep» which is a generalization of «gaAcqStep», «gaRelStep», «saStep» stereotypes. So, any changes in the structure of the attributes of «gaScenario» imply changes in the attributes of «gaStep» and those of its various specializations (Figure 9 and Figure 10).

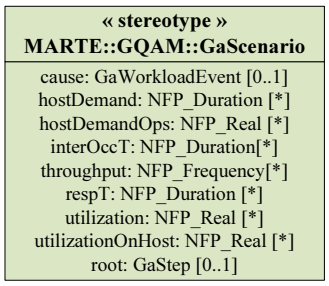

Figure 9. the basic structure of «GaScenario»

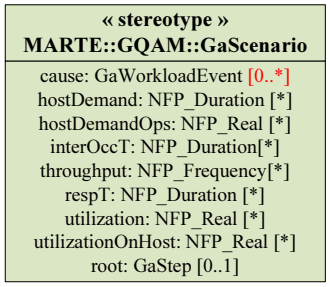

Figure 10. «GaScenario» meta-model with changes made

VI. Pedagogic case study

To better explain our contribution, we rely on a pedagogical example. We consider a simple system with an application made by three periodic tasks and a target architecture holding two heterogeneous computing resources. For the tasks scheduling stage, we make use of PFair which is an on-line multiprocessor scheduling algorithm, with dynamic priority, and a part of the global approach. It enables tasks and instance's migration. We apply it in the context of periodic and interrupted tasks such as the deadline is equal to the period. Actually, in the present example, we will not deal with instance's migration, but we will rather focus only on task's migration to facilitate understanding our contribution. The schedulability modeling based on MARTE does not use only the sub-profile SAM but also some attributes of the GRM package to annotate a class diagram in order to clarify the different concurrent software resources and those of the target architecture. Figure 11 specifies a class diagram annotated by MARTE/GRM stereotypes. The figure 12 illustrates in a better way the content of a TASK class and the value of important attributes of «swSchedulableResource». It is a zooming for the TASK class illustrated in Figure 11. The use of SAM stereotypes within the schedulability analysis context is validated on an activity diagram. First, we start by specifying the GRM attributes required to support a multiprocessor scheduling as part of a global approach. Then, we validate our contribution while handling an activity diagram annotated by SAM.

A. GRM view and multiprocessor schedulability analysis with task's migration

The stereotype «swSchedulableResource» annotates a software concurrent resource. Among the attributes used in the schedulability analysis, we find the attribute «isStaticSchedulingFeature». This is a Boolean attribute used to specify whether the scheduling parameters are static or dynamic. In our case, we have chosen to use a scheduling algorithm with dynamic scheduling parameters. So, it is mandatory to put the value of «isStaticSchedulingFeature» on false. Furthermore, to specify whether the scheduler allows preemption of tasks, it is required to set true or false in the Boolean attribute «isPreemptible». This is an attribute of «scheduler» stereotype.
As noted below, the GRM annotations are applied on the class diagram exposed by Figure 11. Indeed, this view includes a set of operations such as the task's activation, its preemption, etc. Given the dynamic aspect of the chosen scheduling algorithm, a task's priority changes according to a set of criteria. Afterward, we choose to add a function of the type integer in our class diagram. This function computes the priorities of tasks. It is the function «computingPrio ()». It will be denoted by the stereotype «gaRequestedService» and «swAccessService». Through the attribute «swAccessService», we specify that the priority is dynamic. Therefore, we attribute the true value to «isModifier». «accessedElement» must be equal to the property which defines the priority in the class TASK. This is the property «priority». We also need to specify through the attribute «owner» the name of the resource which owns the priority. These are the various tasks. We will later see that the annotation by «gaRequestedService» is necessary to call the function «computingPrio ()» in the activity diagram. Moreover, since we use a scheduling algorithm which is not supported by MARTE, we need to attribute the value «other» to the «schedPolicy» attribute, and then, in the attribute «otherSchedpolicy», we put the used scheduling algorithm name. For example, in our case «otherSchedPolicy» = PFair. It should be noted that «schedPolicy» and «otherSchedPolicy» are attributes of the «scheduler» stereotype.

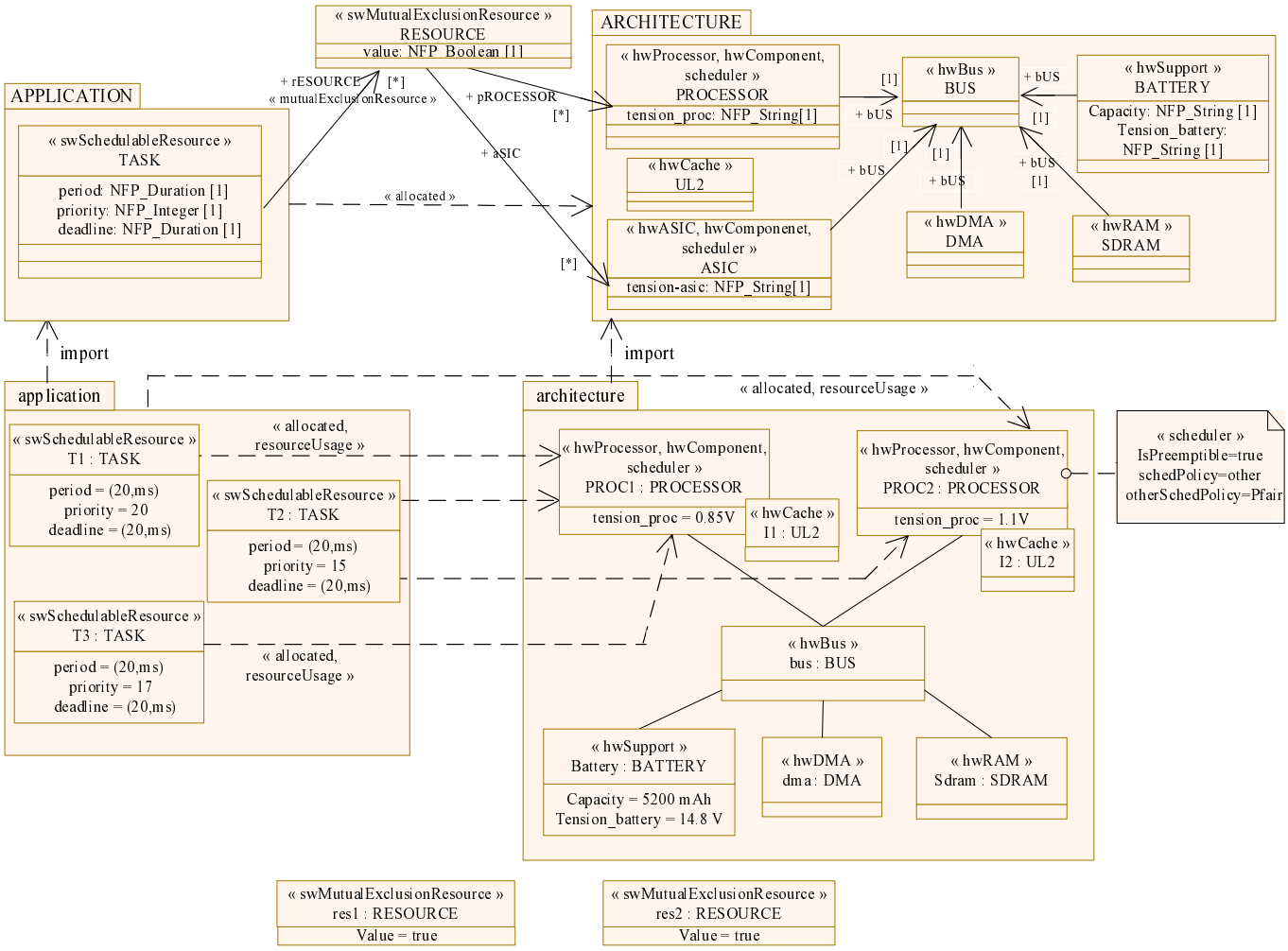

Figure 11. GRM view of the studied system

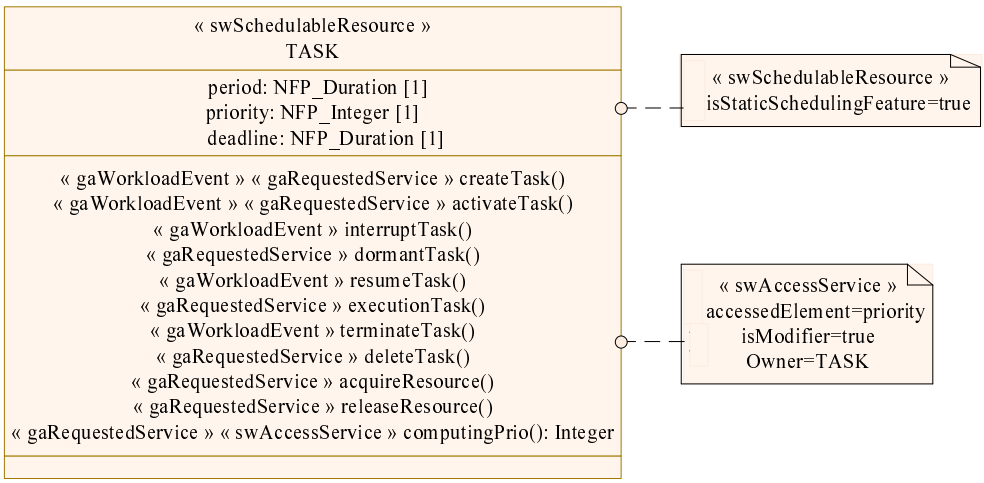

Figure 12. GRM view of a Task

B. SAM view and multiprocessor schedulability analysis with task's migration

In this subsection, we set up the activity diagram (Figure 14) corresponding to our contribution and to the example of the studied system's scheduling. We first specify through the Figure 13 the sequence of tasks scheduling on the available resources for two periods. As noted below, in this example, we deal only with task's migration without handling with instance's migration. This explains the fact that the processor is free for a few moments.

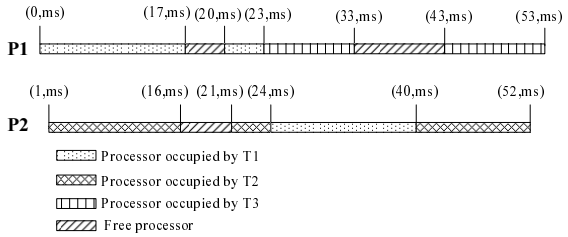

Figure 13. execution order of tasks on available processors

The task's priority is dynamic. Indeed, the aim of this paper is not to deal with the change of priority. But to explain the strategy envisaged, we assume that every non-satisfaction of the demand for locking a shared resource the task's priority increases. Moreover, at every moment of shared resource liberation, the task's priority increases when the concurrent resource is interrupted. It retains the same value if the task has finished executing. We try to explain the activity of Task T1 presented in the activity diagram (Figure 12). The task is periodic with a period=20ms and an initial priority («priority» = [17]). To begin, the task must allocate a shared resource that is designed through the action LockResource annotated by «gaAcqStep». The locked resource is res1 and the corresponding processor is proc1, which is specified by «acqRes» and «host» attributes (acqRes= [res1] and host= [proc1]), respectively. The first request of blocking resource is caused by the T1 activation at t=0ms. Once the processor is allocated, T1 will be executed. Its running time is 17ms. For this period of time, the task will never be interrupted, which is specified via the attribute «preemptT» which is equal 0 ms. Thus, the release of the shared resource, provided by the action UnlockResource annotated «gaRelStep», is due to the termination of the task's execution. This condition is specified through the attribute «cause» by calling the operation terminateTask () (cause= [terminateTask]).

As mentioned above, the release of a shared resource involves changing the priority if the task is interrupted and if the priority is not conserved. Then, the UnlockResource action calls the computingPrio() via the «servDemand» attribute. (i.e. the computingPrio() operation is called for each locking and unlocking of a shared resource). The result of this function is stored in the attribute «priority». Since T1 was not interrupted, after releasing the shared resource, it retains the same priority. Then, the attribute «priority» of the «gaRelStep» stereotype is set to 17 (priority= [17]). Note that the attribute «concurRes» indicates the name of the currently-running task's instance.

For a new period, T1 locks a shared resource. This time, the blockage is caused by «resumeTask». This is an operation that involves the reactivation of the task. The locked shared resource is res1. At this level, acqRes= [res1, res1] (the first res1 was locked during the first period and the second one is currently locked). The blocking request is satisfied. Therefore, the task remains the same priority 17. Then, the attribute «priority» of «gaAcqStep» is set to priority= [17, 17]. The corresponding processor of res1 is proc1, so host= [proc1, proc1]. The processor proc1 is already allocated, the task proceeds to be executed. It begins at t=20ms, which is annotated through the attribute «readyT» which takes the value 20ms (readyT= [0ms, 20ms]). T1 runs for 3ms before being interrupted. At this level, execTime= [17ms, 3ms]. T1 is interrupted by a higher priority task T3. It should be noted at this level that it releases the shared resource res1 that is specified through the action UnlockResource. At this stage, relRes= [res1, res1]. As can be noticed, this liberation is caused by an interruption. So, the attribute «cause» takes the value interruptTask (cause= [terminateTask, interruptTask]). Furthermore, since the task is interrupted, the priority increases and will be 18, and the content of the attribute priority becomes (priority= [17, 18]). Note that the priority is equal to 18 as regards the stereotype «gaRelStep». T1 will not be interrupted pending a free computing resource, but will rather interrupt a lower task which is T2. Next, the processor proc2 will be available to the one to which T1 will migrate. Normally, in this case interruptT=0ms, yet, we assume that the context switch time is equal to 1ms. This period of time will be indicated as a preemption time. Then, this value will be added in the attribute «preemptT». Therefore, at this level, the content of «preemptT» becomes preemptT= [0ms, 1ms]. T1 will continue its running on proc2. Its priority is 18 and «priority» attribute becomes priority= [17, 17, 18]. First, T1 proceeds to lock the shared resource res2 corresponding to proc2. Therefore, res2 will be added to the attribute «acqRes» of the stereotype «gaAcqStep», as well as proc2 will be added to host (host= [proc1, proc1, proc2]). Since the job T1 will continue running, we add «resumeTask» to the attribute «cause». Then, the content of «cause» will be cause= [T1Created, resumeTask, resumeTask]. Subsequently, T1 is considered to be executed. It will run for 16ms. This value will be added in the attribute «execTime» of the stereotype «saStep». At this level, execTime= [17ms, 3ms, 16ms]. T1 ends its running on proc2 without being interrupted. So preemptT= [0ms, 1ms, 0ms].

In consequence, T1 frees the shared resource res2 within cause= terminateTask. Then, at this step, the attribute

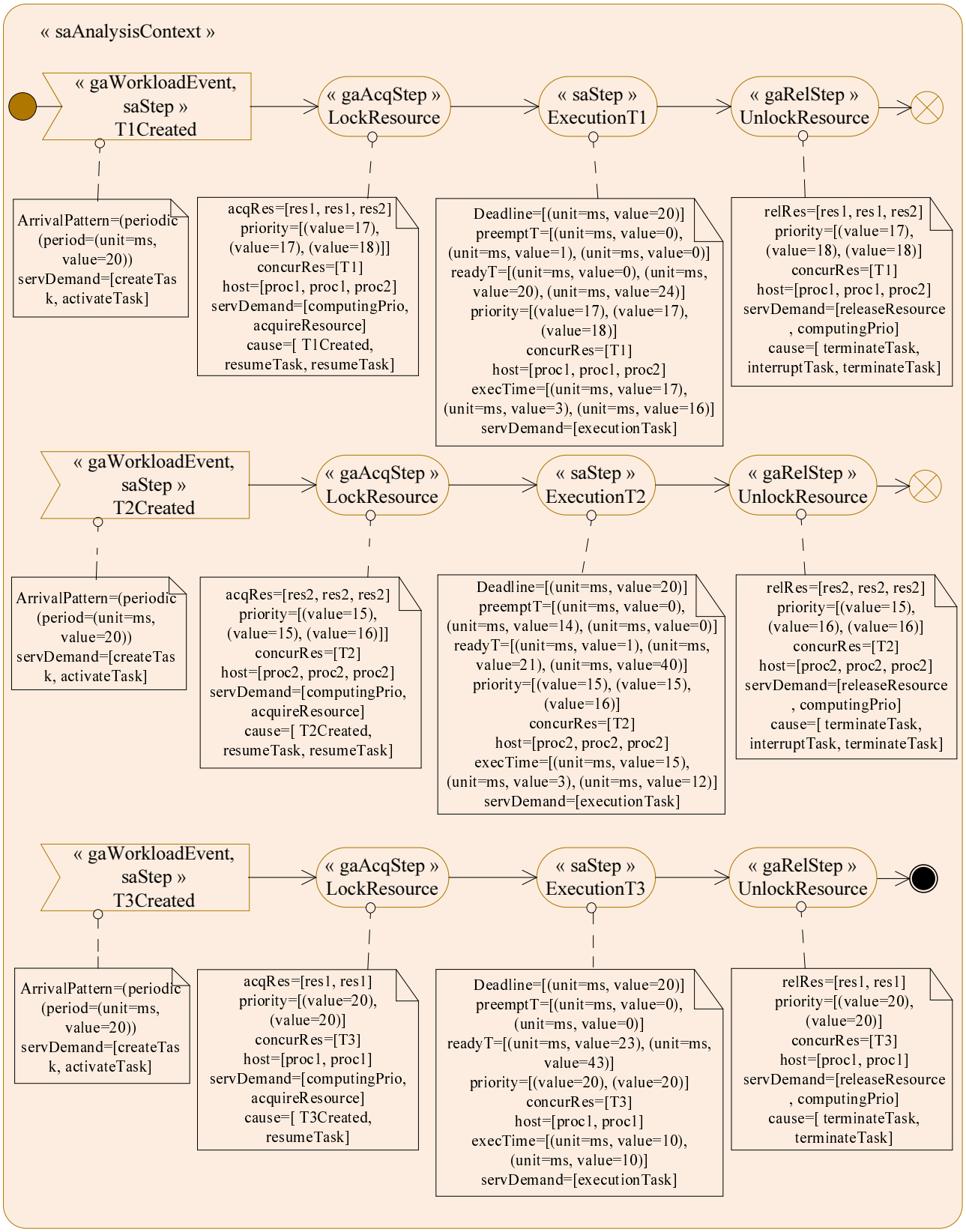

Figure 14. activity diagram annotated through SAM stereotypes

«cause» of the stereotype «gaRelStep» will be cause= [terminateTask, interruptTask, terminateTask]. The task T1 ends with the same priority 18. So, the «priority» attribute of the stereotype «gaRelStep» will be priority= [17, 18, 18].

VII. Conclusion

Throughout this paper, we have proposed an extension to MARTE profile since it does only support scheduling algorithms for the partitioned approach. This improvement affects mainly the sub-profile MARTE/GQAM and especially GQAM/SAM. It accounts for the multiplicities of some existing attributes. This contribution makes MARTE able to stand the multiprocessor scheduling algorithms for the global approach. As future works, we seek to make a formal scheduling analysis.

References

[1] Saoussen Anssi, Sara Tucci-Pergiovanni, Chokri Mraidha, Arnaud Albinet, François Terrier, and Sébastien Gérard. Completing east-adl2 with marte for enabling scheduling analysis for automotive applications. In *Conference ERTS 2010, Toulouse*, 19 to 21 May 2010.

[2] Sanjoy Baruah and Joël Goossens. Scheduling real-time tasks: Algorithms and complexity, 2003.

[3] Sébastien Demathieu and Laurent Rioux. Marte to rapidrma. Thales Report/technical Document number 61565273 305 6.

[4] Sébastien Gerard, François Terrier, and Yann Tanguy. Using the model paradigm for real-time systems development: Accord/uml. In *OOIS '02: Proceedings of the Workshops on Advances in Object-Oriented Information Systems*, pages 260–269, London, UK, 2002. Springer-Verlag.

[5] Joël Goossens. *Introduction à l'ordonnancement temps réel multiprocesseur*, pages 157–166. 2007.

[6] OMG Object Management Group. Uml profile for schedulability, performance and time. 2002.

[7] OMG Object Management Group. A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, Beta 2, ptc/2008-06-09. Object Management Group, June 2008.

[8] J. Javier Gutiérrez, José M. Drake, Michael González Harbour, and Julio L. Medina. Modeling and schedulability analysis in the development of real-time distributed ada systems. *Ada Lett.*, XXII:58–65, April 2002.

[9] 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*, 11-13 April, 2012.

[10] Matthias Hagner and Michaela Huhn. Tool support for a scheduling analysis view. In *Workshop "'Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile"' at DATE'08 (Design, Automation & Test Europe)*, 2008.

[11] Yessine Hadj Kacem, Walid Karamti, Adel Mahfoudhi, and Mohamed Abid. A petri net extension for schedulability analysis of real time embedded systems. In *The 16th International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA'10*, pages 304–314, 2010.

[12] Joseph Y T Leung and Jennifer Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2(4):237–250, 1982.

[13] C L Liu. Scheduling algorithms for multiprocessors in a hard real-time environment. *JPL Space Programs Summary*, 1969.

[14] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[15] Julio Medina and Alvaro Garcia Cuesta. Experiencing the uml profile for marte in the generation of schedulability analysis models for mast.

[16] Julio L. Medina and Alvaro Garcia Cuesta. Model-based analysis and design of real-time distributed systems with ada and the uml profile for marte. In *Proceedings of the 16th Ada-Europe international conference on Reliable software technologies*, Ada-Europe'11, pages 89–102, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] M.-A. Peraldi-Frati and Y. Sorel. From high-level modelling of time in MARTE to real-time scheduling analysis. In *MoDELS'08 W. on Model Based Architecting and Construction of Embedded Systems on ACES-MB*, pages 129–143, Toulouse, France, September 2008.

[18] Douglas C. Schmidt. Model-driven engineering. *IEEE Computer*, 39(2), February 2006.

[19] Lui Sha, Tarek Abdelzaher, Karl-Erik Arzen, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysious K. Mok. Real time scheduling theory: A historical perspective. *Real-Time Systems Journal*, 28(2/3):101–155, 2004.

[20] François Terrier and Sébastien Gérard. Mde benefits for distributed, real time and embedded systems. In *DIPES*, volume 225 of *IFIP*, pages 15–24. Springer, 2006.

[21] Martes Workshop, Huascar Espinoza, Julio Medina, Hubert Dubois, Sébastien Gérard, François Terrier, and Cea-list Dtsi S O L L-lsp. Towards a uml-based modeling standard for schedulability analysis of real-time systems. *October*, 2006.