Using MDE and Priority Time Petri Nets for the schedulability analysis of Embedded Systems modeled by UML activity diagrams

Yessine Hadj Kacem*, Amina Magdich*, Adel Mahfoudhi*, Walid Karamti*
and Chokri Mraidha†
*University of Sfax, ENIS, CES Laboratory
Soukra km 3,5 B.P.: 1173-3000 Sfax TUNISIA
Email: {yessine.hadjkacem, amina.magdich, adel.mahfoudhi, walid.karamti}@ceslab.org
†CEA, LIST, Gif-sur-Yvette
F-91191, France
Email: chokri.mraidha@cea.fr

***Abstract*—This paper proposes a model driven approach for the schedulability analysis at an early stage of the embedded system development life-cycle. The activity diagram of Unified Modeling Language (UML) annotated with the profile for the Modeling and Analysis of Real-Time and Embedded systems (MARTE) is mapped into Priority Time Petri Net (PTPN) to enhance formal schedulability test of given real time tasks. The generated PTPN model is interpreted and executed to check whether a schedule of a task execution meets the imposed timing constraints.**
Therefore, the present paper focuses on the definition of temporal properties and tasks dependency by means of activity diagram and MARTE profile. Besides, it describes the transformation rules from analysis model to formal model.

***Keywords*-activity diagram; MARTE; mapping rules; PTPN; scheduling analysis**

I. INTRODUCTION

The Model Driven Engineering (MDE) [13] is seen as very effective in dealing with the real time embedded system complexity. It aims at decreasing the growing complexity of real time systems and verifying their correctness. Hence, the verification of system properties particularly real time and precedence constraints is an essential issue. One of the most important challenges is the schedulability test of given real time tasks. In fact, the scheduling analysis phase makes it possible to predict and validate the system temporal behaviour before its realization. Thus, it allows the designer to face the development risks. It must be certainly included during the judgement and validation of each possible implementation. An analysis carried out earlier makes it possible to guide the search for a better Software/Hardware implementation during the development cycle. As a consequence, the analysis of non-functional properties has been integrated at a high level abstraction layer in real systems development life cycle.

UML profiles promote an adequate solution to represent different system views with their embedded and real time features. The recent MARTE profile [3] adopted by the Object Management Group OMG, fosters the building of models that support the specification of scheduling analysis problem. Though it is a powerful and advanced standard for annotating models with the required information for performing scheduling analysis, it does not provide a technique for verifying models. The UML extension lacks a tool to check system properties and constraints. So, a need emerges for mapping used models to an external platform for schedulability test. Talking about scheduling analysis methods [14], in literature, allows us to distinguish between three techniques that are simulation, feasibility test and model checking. With regard to simulation, it is based on the execution of the task progress during one system period, which assures that all the authorities of tasks respect their deadline. It can treat scheduling policies or task models that are difficult to analyze mathematically. As for feasibility test that applies a formula, it decides the realizability of task scheduling according to which characteristics appear to be the simplest method to implement with a low calculation complexity. Contrary to the simulation and test of feasibility, the model-checking provides examples showing why an imposed constraint is not satisfied. In this context, some research studies [12] [10] [6] [9] aim at transforming UML/MARTE model into a scheduling analysis tool such as [4], [2], [16] and [15]. Unfortunately, UML diagrams are generally mapped into simulation platforms for schedulability test as mention. It should be recalled that the application of formal analysis techniques of UML views has become a very active field of research in the last years. Although, a few methods tackle the transformation of UML activity diagrams annotated with MARTE profile into formal models, only event's flow has been treated, some elements of activity diagrams will not be processed, which leads to the impossibility of transforming complicated UML activity diagram into Petri Nets.

In the same vein, Mallet et al. have introduced in [8] mapping rules of an activity diagram based on UML to Petri Nets. Although this extension supports real-Time systems, it neglects other points of interest. In other words, the use of UML without benefitting from its profiles (such as MARTE) cannot provide high-level diagrams. So, the

proposed approach in [1] has introduced the transformation rules of an activity diagram based on the adaptation of SysML and MARTE to an extension of Petri «ETPN» (Time Petri Net with Energy Constraints). The adaptation of these two profiles supports the annotation of quantitative properties. ETPN extension can handle temporal and energetic constraints of Real Time Systems. The paper presented in [17] has introduced the transformation rules from an activity diagram UML/MARTE to the Petri extension TCPNIA (Timed Colored Petri Nets with Inhibitor arcs). In brief, the various proposed extensions of Petri Nets cited in the previously mentioned work did neither support the problem of scheduling analysis nor address the conflict when crossing transitions.

Our proposal differs from the aforementioned ones in the sense that it includes the application of UML diagram and the MARTE profile and formal verification for the schedulability test of given tasks and their implementation. Indeed, among the existing formalisms, we bet our choice on Priority Time Petri Nets (PTPN) [5] tool due to its special sufficiency to support the scheduling problem. Moreover, the priority included in this extension deals with the conflict during the crossing transitions. Furthermore, its hierarchical composition allows the considerable reduction of the size and complexity of the nets[7].

Hence, the approach presented here adopts the model driven engineering. Indeed, it takes a UML2 model instance captured through the Activity diagrams annotated with MARTE profile stereotypes. A particular model of periodic and dependent real time tasks is adopted. After that, the mapping from UML into PTPN is performed. Next, the obtained network is executed to verify whether a schedule of a task execution meets the imposed timing constraints. The results supply a description of the temporal fault to help the designers in refining the partitioning space Sw / Hw. Yet, the process goes to the methodology starting point by changing the allocation of the task on an execution resource. After an adjustment of tasks implementation, the adopted process is executed again.

The remainder of this article is organized as follows. Section 2 provides an overview of the main concepts of PTPN. As an introduction to the technical sections, basic concepts on UML activity diagram and MARTE profile are introduced in section 3. Then, Section 4 presents the used process to translate a UML activity diagram into a PTPN model. In section 5, we tell our experience from a football player robot application in which our method has been applied. Finally, the proposed approach is briefly outlined and future perspectives are given in the last section.

II. Overview of PTPN

PTPN is a 3-tuple defined by $PTPN =< R, T_f, P_r >$ where:

1) R is a regular Petri Net
2) $T_f : T \mapsto \mathbb{Q}^+$ is the firing time of a transition
3) $P_r : (T \times P) \mapsto \mathbb{N}$ is the priority of a transition according to a place

In what follows, the PTPN model construction is explained and the notion of PTPN component is then stressed.

A. PTPN Model Construction

This subsection illustrates how PTPN is used to model real time tasks and especially their periodicity, priority, dependency and distribution on a distributed architecture. In fact, each task is characterized by a period, a deadline and an activation date. These features are represented by «Period», «Deadline» and «TiCreated» places as well as «T-Period», «Activation» and «T-deadline» transitions, respectively. To be executed, a prior task must lock a free processor so that it can block a mutual exclusion resource through the «BlockingRes» transition. Subsequently, the processor will be blocked after the crossing of «BlockingProc» transition. The crossing of «TakingProc» transition causes the recovery of the computing resource. A task can send or receive data to/from other concurrent tasks, which is a dependency that is represented through «sendData» and «Ti-receive» transitions, respectively. A PTPN model for Task T1 is illustrated by Figure 1. More details about the specification of the scheduling analysis problem can be found in [5]

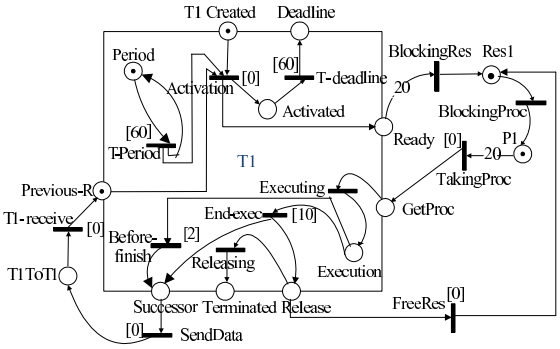


Figure 1. Task modeling with PTPN

B. Local PTPN components for scheduling Analysis specification

A Real Time Embedded System (RTES) consists of a set of real-time tasks and a set of processors. The formal definition of an RTES is presented by definition 1.

Definition 1

The RTES Ω is defined by the 5-uplet $\Omega = \{TK,\ R_s,\ Proc,\ alloc,\ Prio\}$, with:

- TK: is a finite set of real-time tasks with each task determined by the following parameters:
  - R_i: the date of the first activation
  - P_i: the period associated with the task
  - C_i: the execution period of the task for the P_i period

  - D_u: the life cycle of the task (the duration of total execution). For each task, two invariants must be respected: $C_i \leq D_u$ and $R_i \leq P_i$
- $Proc$: a finite set of processors.
- $R_s : TK \mapsto \{TK\} \cup \{\oslash\}$, a function which initializes precedence relations between tasks
- $Alloc : TK \mapsto Proc$, a function which allocates a task to a processor
- $Prio : TK \times Proc \mapsto \mathbb{N}$, a function which allocates priorities to the tasks according to the processor.

The PN paradigm in objects necessitates the encapsulation of the various behaviors of the object in a centenary called PN component. A PTPN component called «Tc» is used to hide the behavior of a real-time task. It shows only the binding of tasks on computing resource as well as communication between tasks. In fact, a PTPN component is composed of an input interface containing input places and an output interface containing output places; so that, mapping UML activity diagram into PTPN component builds on the foundation of the task object as well as its encapsulated attributes. In this way, it consists in the creation of the Processor object, the allocation of tasks on the computing resources and finally the dependency management between tasks. A PTPN component for a Task T1 is presented in Figure 2.

C. PTPN and Model Driven Engineering

The PTPN tool takes the form of a Petri Net editor and an executer model of the modelled net. Indeed, the developed editor of our PTPN relies on the Graphical Modeling Framework (GMF) founded on Eclipse Modeling Framework (EMF). It is obvious that the created model is built around a drawing composed of places, transitions and arcs. The created model can also be serialized to generate an XML (Extensible Markup Language) or XMI (XML Metadata Interchange) file. The generated file conforms to the PTPN Meta Model and presents the entry port point of the executer. Due to the structure of the editor output, the properties of the modelled net are easily interpreted. The verification framework is sufficiently flexible and expressive to support module inclusion and extension.

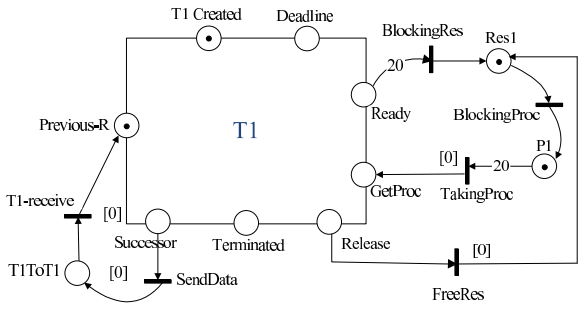


Figure 2. PTPN Task component

III. UML ACTIVITY DIAGRAM AND MARTE NOTATIONS FOR RTES SCHEDULING ANALYSIS

MARTE includes the Schedulability Analysis Modeling (SAM) Modeling package that provides a rich terminology to support early schedulability analysis. Several diagrams can be adopted during the temporal analysis of the system with UML views annotated by SAM package. The activity diagram, in particular, is able to expose the dynamics of a system because it models the sequences of its activities (workflow). Indeed, they are capable of clarifying rather complicated uses cases. They also present a great degree of resemblance to the Petri Nets.

At this level, we focus on the SAM annotations applied to the activity diagram. In fact, an external event depicting an activation event that triggers the behavior of a system will be annotated via the stereotype «GaWorkloadEvent». The latter describes events that can occur repetitively. A periodic event is annotated with the attribute «Arrival Pattern» in order to fix its period. However its launching time will be stereotyped by the attribute «TimedEvent». Moreover, any activity/event which represents the execution of an operation and asks for an execution resource (such as a processor) will be annotated by «saStep». Through its attributes, it allows the specification of the execution time via «exec-Time». The priority, the deadline and the execution resources are mapped to «priority», «deadline» «Host» respectively.

Moreover, «saStep» has the same attributes: «readyT» representing the execution beginning time of a software resource, «preemptT» specifying the duration of pre-emption of a task and «concurRes». «concurRes» allows the specification of the concurrent resource in the course of execution on the considered resource calculation. As it is possible to send or receive data between various software resources, it will be necessary to be based on a SAM stereotype ensuring the annotation of such an action. In fact, the related stereotype is «saCommStep». It is characterized by a set of attributes allowing a varied annotation characterizing the sending or the reception of a message. In particular «respT» specifies the required time interval in order to send or receive data. To be executed, a software resource must obviously be allocated to a resource of hardware or software execution. Consequently, it can, if necessary, block a mutual exclusion resource shared to prohibit the access to the required processor. This blocking is done via the stereotype «gaAcqStep» which allows the specification of the name of the mutual blocked resource via the attribute «acqRes» and the time necessary to block it through the attribute «respT». The task end execution on a previously-blocked computing resource automatically causes its release following the release of the mutual resource which protects it. It is annotated by the stereotype «gaRelStep». Through its attributes, «gaRelStep» allows to state the name of the mutual exclusion resource to release through «relRes» and

the time necessary to be performed via the attribute «respT». The allocation, the execution of the set of tasks on the various processors and the communication between these various software resources are represented by the activity diagram presented in Figure 3. It is worth while to mention that the given example deal with a real case study. More details about it will be given in the last section.

IV. Mapping Strategy

The direct specification of scheduling analysis problem within PTPN formalism remains a tedious task which requires an expert knowledge. Compared to UML, the established PTPN model contains more nodes and edges than UML activity diagram. It is also difficult to understand, read and maintain. Activity diagrams annotated with MARTE profile stereotypes is based on abstraction, separation of concerns. It can cover a great part of the life cycle complex RTES design with their real time constraints and performance issues. However, UML lacks a tool to check system properties and constraints, the mapping of used model to a test schedulability tool is necessary. The rest of this section depicts how to extract the elements from MARTE activity diagram including timings constraints. The method consists of the deriving activity diagram elements (nodes, transitions, signals, actions, and synchronisation bar) and MARTE annotations into PTPN elements. Nevertheless, we opt for the translation of the source model into PTPN components.

A. Mapping UML activity diagram to PTPN elements

Mapping Nodes

An initial node corresponding to the launching of the scenario of the system will be transformed into a place marked by an initial token. As for the node of the flow termination, it expresses the end of an action regardless of the end of a system activity. However, the final node indicates the system end scenario. In the same way it will be transformed towards an empty place without output arc. Table I illustrates the process of the transformations of the initial and final nodes.

Mapping Transitions

In UML, a transition is a bond between two actions which can take place only if the preceding activity has already been finished, it will be transformed towards a Petri transition.

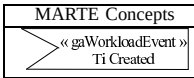
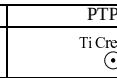
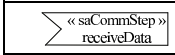
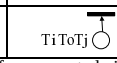
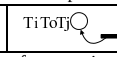
Mapping Signals

Table I
MAPPING NODES

MARTE Concepts	PTPN
	
Transformation of an initial node	
	
Transformation of a final node	

An accepted signal represents the arrival of a signal. Of course, its transformation into PTPN depends on the SAM stereotype that annotates it. The accepted signal «Ti Created» annotated by «gaWorkloadEvent» indicating the arrival of a periodic software resource will be transformed into a place marked by only one token. However, the accepted signal «receiveData» stereotyped by «saCommStep» expressing the reception of a data sent between two dependent tasks or within the same task will be transformed into a place and a transition connected by an input arc.

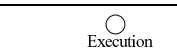
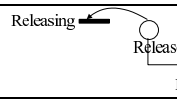
Table II
TRANSFORMATION OF SIGNALS

MARTE Concepts	PTPN
	
	
Transformation of an accepted signal	
	
Transformation of a sent signal	

The sent signal «sendData» annotated by «saCommStep» indicating the data sent between two dependent tasks or within the same task will be transformed into a place connected to a transition through an output arc. Table II presents a summary of the transformation process of the two signals types.

Mapping Actions

Table III
TRANSFORMATION OF ACTIONS

MARTE Concepts	PTPN
	
	
	

The action «Lock_Resource» annotated by «gaAcqStep» is prerequisite if the access to the computing resource requires its blocking. Consequently, as its stereotype, this action indicates the blocking of a mutual exclusion resource and the recovery of the corresponding execution resource. The corresponding transformed PTPN is a place connected to a transition by means of an output arc. The task execution on the computing resource is presented by the action «Execution Ti» stereotyped by «saStep». This action with the related stereotype will be transformed into a place without any token at the beginning. The end of the execution automatically causes the release of the blocked mutual

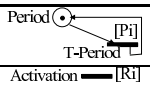
exclusion resource as well as the related computing resource. The action «unlock_Resource» annotated by «gaRelStep» is mapped into a place attached to two transitions via two arcs as shown in Table III.

Transformation of the attributes of stereotypes

The attribute «arrivalPattern» indicates that the task is periodic and that its period is fixed. The periodicity will be transformed into a place connected to a transition via an input arc and another output one. The value of the period will be represented by a crossing condition of a transition. Moreover, the attribute «TimedEvent» representing the action activation will be transformed into a temporal firing condition of a transition.

«acqRes» is an attribute allowing the specification of the name of the blocked shared resource. It will be transformed towards a marked place. The blocking of this resource is characterized by a fixed response time set via the attribute «respT» which will be transformed into a temporal firing condition of a transition.

Table IV
TRANSFORMATION OF THE ATTRIBUTES OF STEREOTYPES

MARTE Concepts	PTPN
«arrivalPattern»	
«TimedEvent»	Activation [Ri]
Transformation of the attributes of «gaWorkloadEvent»	
«acqRes»	NameRes
«respT»	BlockingRes [Ti]
Transformation of the attributes of «gaAcqStep»	
«Deadline»	Deadline [Ti] T-deadline
«execTime»	End_exec [Ci]
«priority»	prio
«Host»	NameProc
Transformation of the attributes of «saStep»	
«relRes»	NameRes
«respT»	FreeRes [Ti]
Transformation of the attributes of «gaRelStep»	
«respT» for data sending	SendData [Ti]
«respT» for data receiving	Tj-receive [Ti]
«readyT»	Before-finish [Ti]
Transformation of the attributes of «saCommStep, saStep»	

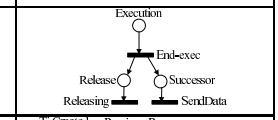
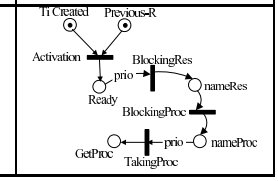
As for «saStep», it is founded on certain attributes making possible the integration of non-functional properties. . Let us start with the temporal attributes which arise in «execTime». They allow the indication of execution time that puts a task during its execution on a given processor, it will be transformed into a temporal condition. Thus, the attribute «Deadline» will be translated into a place attached to a transition through an output arc. However, its value will be transformed into a temporal condition of transition's crossing. Then, the attributes «priority» and «Host» respectively expressing the priority of a task compared to the concurrent tasks and the name of the allocated processor will be mapped to «Prio» and «nameProc».

The stereotype «gaRelStep» allows the release of the previously blocked mutual exclusion resource. It has similar attributes «relRes» indicating the name of the released resource and «respT» specifying the required time to be done. These concepts will be mapped to «NameRe» and the temporal condition of the firing transition «Free-Res».

The «saCommStep», indicating the sending or reception of data, has the same attribute «respT» specifying the necessary time to send or receive a signal. The transformation of this attribute corresponds to a temporal crossing condition of a transition. In fact, if the task sends the message before the end of its execution, this moment will be fixed via the attribute «readyT» related to «saStep». Consequently, we need to gather the two stereotypes «saCommStep» and «saStep» in order to benefit from the attribute «readyT». This attribute will be transformed into a condition. Table IV illustrates the various transformation phases of the stereotypes that annotate the activity diagram into PTPN elements.

Transformation of a synchronization bar

Table V
TRANSFORMATION OF SYNCHRONISATION BAR

MARTE Concepts	PTPN
The fork	
The synchronisation	

A synchronization bar is a solution to model concurrent processes. In fact, there exist two families of synchronization bars. The first of which is the forks also known under the name of disjunction which represents simultaneous release of several outgoing transitions. The other family is called junction, which describes the synchronization of several transitions in entry. Indeed, in the junction case of an outgoing transition, it can be crossed only if all the transitions in entry are carried out. The mapping of synchronisation bar to PTPN elements is illustrated in Table V. The complexity of transforming the activity diagram into PTPN elements is explained through the big size of the network specifying the problem of schedulability analysis. Therefore, it becomes difficult to understand, manage and particulary to interpret. In order to minimize the processing cost, we choose the translation into PTPN components for a better clear and readable representation.

B. An alternative of the activity diagram formal checking: Transformation into PTPN components

A PTPN component is a kind of container encapsulating the object behavior. In our context, it encapsulates the way to act for real time task. We suggest a PTPN component named «COMP-ptpn» composed of an input interface including the input places and an output interface containing those of output. The input interface represents the input places $Ti_{Created}$, $Previous_R$ *and* Get_{Proc}. However, the output interface is composed of the ouput places «Deadline», «terminated», «successor», «Release». Thus, the mapping strategy covers mainly the task object creation and its attributes, the processor object creation object processor and finally the dependence management.

Transformation into Task object

The initial state of the activity diagram that triggers the system scenario as well as the accepted signal indicating the task creation will be transformed into an input marked place. The termination node of the flow or final node indicating the end scenario of a task and the system end scenario respectively will be translated, in the same way, into a place of the output interface.

It is worthy to note that each element of the encapsulated PTPN specifies a task temporal characteristic as its period, its activation date, its deadline and its execution time. It will be produced as it is indicated in Table VI.

Transformation into Processor

Table VI
TRANSFORMATION OF TASK CHARACTERISTICS INTO ENCAPSULATED PTPN ELEMENT

MARTE Concepts	PTPN encapsulated
«SaStep»	PTPN.P.execution
«arrivalPattern»	PTPN.P.Period, PTPN.T.T-Period, PTPN.Tf(T-Period), PTPN.ARC.inputArc, PTPN.ARC.outputArc
«TimedEvent»	PTPN.T_f.T_f(Ativation)
«execTime»	PTPN.T_f.T_f(End-exec)
«readyT»	TPN.T_f.T_f(Before-finish)
«Deadline»	PTPN.P.Deadline, PTPN.T.T-deadline, PTPN.ARC.outputArc, PTPN.T_f.T_f(T-deadline)

A processor modeled via the attribute «Host» of the stereotype «saStep» will be transformed into a place «NameProc».

Communication Task/ Processor management

The communication task/ processor is translated via the requirement and release of the processor during a certain period. In fact, any activated task requires a computing resource on which it will be executed. The recovery of such resource is conditioned by the resource availability and the priority of a task compared to its competitors. When these two conditions are satisfied, an activated task passes to block an exclusion mutual resource, if necessary, in order to recover the required computing resource. This blocking is carried out via «gaAcqStep» which will be transformed into a place and a transition connected via an outgoing arc. «acqRes» is an attribute of «gaAcqStep» which specifies the name of the blocked resource. It will be translated into a place bearing the name of the question resource.

When it finishes its job, a task must release the previously occupied execution resource. So, it is specified through the stereotype «gaRelStep». This annotation will be transformed into a place and two transitions connected between them via two arcs. The attribute «relRes» allows the identification of the released resource name. Consequently, it will be transformed into a place bearing the name of the related resource. The time necessary for the recovery or the release of a shared resource is indicated via the field «respT». Thus, it will be transformed into a transition.

Communication between tasks: dependency concept

Certainly, several tasks can be allocated to the same execution resource. Therefore, they will be dependent one on the other. The trigger of the task execution on the corresponding resource is conditioned by receiving data from the prior task under execution on the required resource. In fact, the dependence between tasks is only conditioned by the requirement of the same execution resource on behalf of various tasks. However, two tasks able to be executed on various processors risk to be dependent. A task can begin its work only after receiving data from another task. The sending and reception of data between the tasks is indicated through the sent signal «sendData» and the received signal «receiveData» stereotyped, in the same way, through «saCommStep» respectively. Each signal will be transformed into a place and a transition connected by an arc. The necessary response time while sending or receiving data will be specified via the field «respT». This attribute will be translated into a temporal crossing condition of a transition.

V. CASE STUDY

This section is devoted to the illustration of the applicability of the proposed approach through a case study. The experiment presents a football player robot application [11] in which video tasks for object detection, wireless communications for message exchanging with other devices, motors controls, sensor acquisition, image processing and decision computation are included. The studied system is composed of four major parts:

- Acquiring and processing image. It is handled through tasks T2, T5, T7, T8 and T9;
- Communication HF: The information exchanges between the robot, players and coaches are made by the following tasks: T1, T4;
- T6 and T12. Knowing that while T12 is used to send data, T1, T4 and T6 serve for reception;
- Data fusion by task T10 and path computation through T11;
- Control of location: it is done through the new trajectory coordinates calculated by the task T11 and through

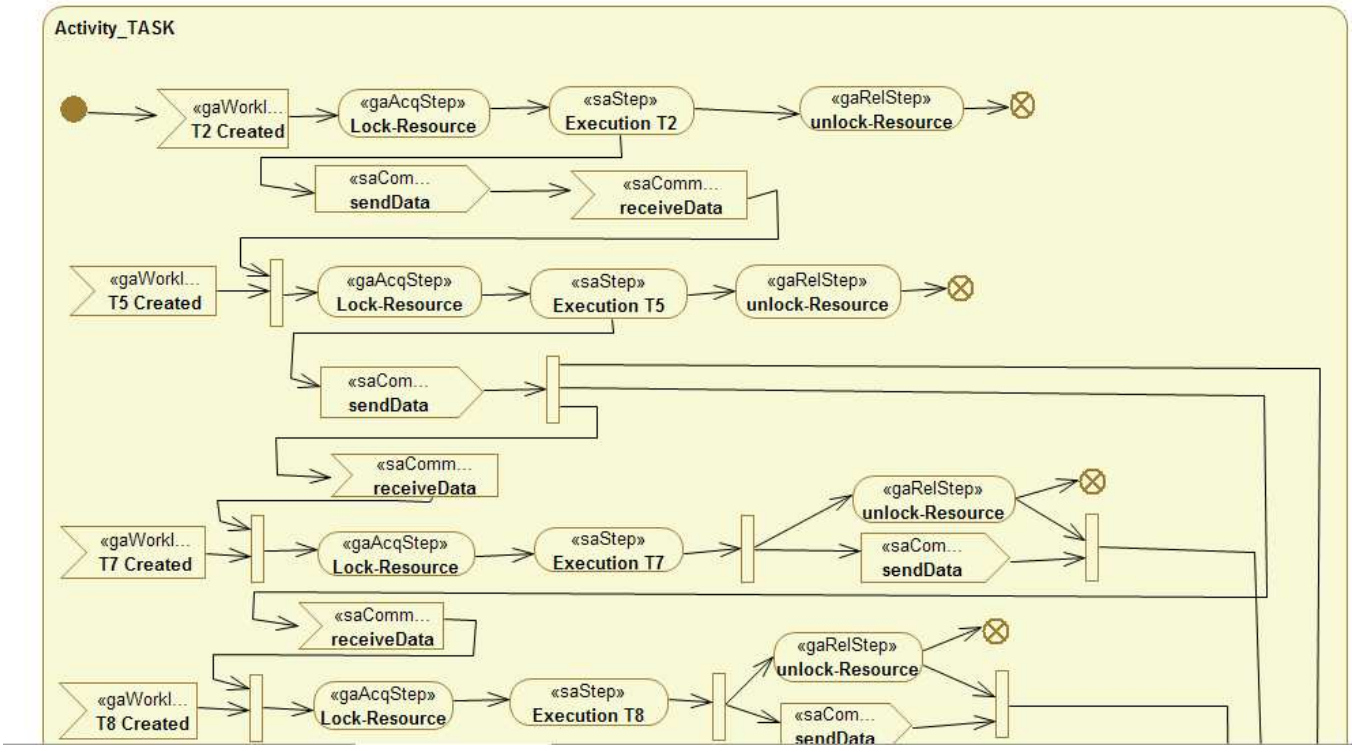


Figure 3. Schedulability analysis via the UML activity diagram annotated with MARTE profile

the current robot position. The location is computed through task T3. Thereafter, T13 controls the motors;

As for the system architecture, it is composed of a processor and three ASICs. In addition, the robot architecture includes a set of memories: cache memory, DMA and RAM. It also covers a battery and a communication bus. Figure 3 shows a portion of UML/activity diagram annotated with SAM stereotypes. The main tasks characteristics and dependency are stressed in this diagram. The dependences between the 13 studies tasks are defined as follows:

- T1 sends data to T4 to convey a message with T6 which can begin its execution on the corresponding processor;
- T2 sends data to T5 to allow its execution;
- T5 sends a message to T7, T8 and T9 to wake them;
- T10 can begin its execution only after the reception of data since T6, T7, T8 and T9;
- After its execution, T10 can send a message to T12;
- T3 becomes ready for execution after the reception of data from T3 and T11;

Based on transformation rules, the considered activity diagram leads to the PTPN components. The construction of the related state graph shows that T11, T12, T5, and T4 can not meet their deadline. In fact, the transition t(Deadline)of T11, T12, T5 and T4 are fired at 44 ms, 62ms, 102ms and 122 ms respectively.

The formal verification of UML models is not sufficient to indicate a temporal fault but also supplies the combination cause of this fault to the designer. As in the present study, the PTPNS indicates that the partition (T4, T5, T11, T12) on the processor is a combination to be neglected in the future iterations of the HW/SW partitioning.

Compared to the existing methods, we deduce that the proposed timing analysis model is represented in an abstract way since early devolvement stages. The resulting model after the mapping process is easy to analyze. If the verification result is negative, the complex design flow is not supposed to be redone which can considerably decrease the development cycle and avoid the violation of real time requirements. In fact, the test identification while verifying temporal properties can lead to a guided implementation/realization of the software system part on a specific execution platform. Indeed, the formal scheduling analysis participate in reducing the presence of errors leading.

VI. Conclusion

The verification of system properties at an early phase is a recent trend in the development of RTES. The model driven engineering facilitates the model annotation with the system characteristics and their execution by mapping them to verification tools. In this paper, we have presented an approach for RTES schedulability analysis within the MDE framework. In this context and as a first step, we have presented the necessary concepts to build an activity diagram annotated with the recent UML MARTE profile stereotypes. Since UML lacks a tool to check system properties and

constraints, the need for mapping used models to an external platform for schedulability test emerges. Hence, we bet our choice on PTPN which is a Petri Nets extension for the verification of RTES timing constraints. Thus, we have presented the various transformation rules leading to PTPN models. Because of the big size of the obtained PTPN model, it remains difficult to understand, manage and interpret. So, for a better, clearer and readable representation, we have chosen to map our activity diagram into PTPN component. The benefits of our approach are threefold: first we can introduce the annotation of UML models with quantitative properties via the MARTE profile. Besides, the used Petri Nets extension of Petri Nets supports the problem of scheduling analysis. It addresses the conflict when crossing transitions. Furthermore, the PTPN hierarchical composition allows the considerable reduction of the size and the Petri Net complexity.

As future work, we would like to propose a multi processor scheduling analysis approach. Moreover, our ultimate objective will be to propose an approach that addresses HW/SW partitioning using MDE and PTPN.

REFERENCES

[1] Ermeson Andrade, Paulo Maciel, Gustavo Callou, and Bruno Nogueira. A methodology for mapping sysml activity diagram to time petri net for requirement validation of embedded real-time systems with energy constraints. In *Proceedings of the 2009 Third International Conference on Digital Society*, pages 266–271, Washington, DC, USA, 2009. IEEE Computer Society.

[2] Elena Fersman and Wang Yi. A generic approach to schedulability analysis of real-time tasks. *Nordic Journal of Computing*, 11(2):129–147, 2004.

[3] 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.

[4] M. Gonzalez Harbour, J. J. Gutierrez Garciia, J. C. Palencia Gutierrez, and J. M. Drake Moyano. Mast: Modeling and analysis suite for real time applications. *Real-Time Systems, Euromicro Conference on*, 0:0125, 2001.

[5] 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.

[6] Yessine Hadj Kacem, Adel Mahfoudhi, Hedi Tmar, and Mohamed Abid. From uml/marte to rtdt: A model driven based method for scheduling analysis and hw/sw partitioning. In *Eight ACS/IEEE International Conference on Computer Systems and Applications AICCSA*, pages 1–7. IEEE Computer Society, May 16-19, 2010.

[7] Adel Mahfoudhi, Yessine Hadj Kacem, Walid Karamti, and Mohamed Abid. Compositional specification of real time embedded systems by priority time petri nets. *The Journal of Supercomputing*, 59(3):1478–1503, 2012.

[8] Frédéric Mallet, Charles André, and Marie-Agnès Peraldi-Frati. From UML to petri nets for non functional property verification. In *Industrial Embedded Systems, 2006. IES '06. International Symposium on*, pages 1–9, Antibes Juan-Les-Pins, October 2006.

[9] 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.

[10] Chokri Mraidha, Sara Tucci Piergiovanni, and Sebastien Gerard. Optimum: a marte-based methodology for schedulability analysis at early design stages. *ACM SIGSOFT Software Engineering Notes*, 36(1):1–8, 2011.

[11] H.Kitano M.Veloso, E.Pagello. Robocup-99: Robot soccer world cup iii. In *Velsoso (Eds.)*.

[12] Ansgar Radermacher, Chokri Mraidha, Sara Tucci Piergiovanni, and Sébastien Gérard. Generation of schedulable real-time component implementations. In *ETFA*, pages 1–4, 2010.

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

[14] 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.

[15] Frank Singhoff, Jérôme Legrand, Laurent tchamnda Nana, and Lionel Marcé. Cheddar : a flexible real time scheduling framework. *ACM Ada Letters journal, 24(4):1-8, ACM Press, ISSN :1094-3641*, November 2004.

[16] Hedi Tmar, Jean-Philippe Diguet, Abdenour Azzedine, Mohamed Abid, and Jean Luc Philippe. Rtdt: A static qos manager, rt scheduling, hw/sw partitioning cad tool. *Microelectronics Journal*, 37(11):1208–1219, 2006.

[17] Nianhua Yang, Huiqun Yu, Hua Sun, and Zhilin Qian. Mapping uml activity diagrams to analyzable petri net models. In *Proceedings of the 2010 10th International Conference on Quality Software*, QSIC '10, pages 369–372, Washington, DC, USA, 2010. IEEE Computer Society.