Discrete Event Simulation of Embedded Systems from Graphical Notations

Fateh Boutekkouk
Department of Computer Science
University of Larbi Ben M'hidi
Oum El Bouaghi, Algeria
Fateh_boutekkouk@yahoo.fr

Mohamed Benmohammed
Department of Computer Science
University of Constantine
Constantine, Algeria
Ben_moh123@yahoo.fr

Abstract— **In this paper, we present our approach for discrete event simulation of embedded systems. In our case, embedded systems are modelled via UML diagrams with some extensions. From UML diagrams a discrete event simulation model is generated. We use the latter model to estimate system performance.**

Keywords-Embedded Systems; UML ; Discrete event simulation, XMI.

I. INTRODUCTION

The ever complexity of embedded systems (ES) [1] design pushes designers to raise the level of abstraction of ES models. We think that the first impetus behind this tendency is to make ES simulation faster. At the other hand, the emergence of the Unified Modeling Language UML as a standard for software visual object oriented modelling can facilitate the modeling task of ES extremely.

UML [2] is a rich graphical specification language, originally, was used in information systems. The use of such graphical notations help designer to understand, capture and analyze the client requirements at early stages of development in a semiformal manner. In its basic form, it is applicable to a wide variety of systems (open language).
Several key attributes of UML are important to embedded systems [5, 6] :

1. A rich set of notations, executable models and semantics suited for modeling different points of view and simulation.
2. Support for state-machine semantics which can be used for modeling and synthesis
3. Support for object-based structural decomposition and refinement.

Thanks to extension mechanisms offered by UML, several profiles have been proposed to tune UML to the context of embedded and real time systems. Contrary to early versions, UML2.x provides new modelling elements to represent hardware concepts as components, ports, interfaces, and signals.

Since UML does not provide a methodology, it is on designer to define his methodology. We think that the Y chart approach [4] is the most appropriate.

The Y-chart approach is a methodology to provide designers with quantitative data obtained by analyzing the performance of architectures for a given set of applications [4]. The Y-chart approach clearly identifies three core issues that play a role in finding feasible programmable architectures: the application the architecture, and the mapping of the application to the architecture.

Performance estimation models are used to try different mappings between the set of application's functional modules and the set of platform components. During these iterations, designers can try different platform customizations and functional optimizations.

The aim of this paper is to define an UML based approach that permits the modeling and simulation of ES following the Y chart approach principles. The paper is organized as follows: section two is devoted to the related work. Our proposed approach for ES modelling and simulation is detailed in section three. Section four puts light on the discrete event simulation principles with some results before concluding.

II. RELATED WORK

With regard to the application of UML to the ES domain, the literature is very rich [9]. However we can mention some pertinent works.

The UML profile for The UML profile for Scheduling, Performance and Time (SPT) [6] aims to define a minimal set of concepts needed for modelling real-time aspects and to analyze the real-time behavior of an application (schedulability and performance).

HASoC (Hardware and Software Objects on Chip) [6] is an approach to the development of embedded systems. HASoC is an object oriented method, which is based on an iterative, incremental lifecycle. The design process, which uses UML notation, begins with the development and validation of a partial, abstract, executable system model, in which the objects are uncommitted to implementation in either hardware or software. This model is then partitioned into hardware and software, on the basis of design constraints, to create a committed model, which is subsequently mapped on to a system platform, and evaluated against design constraints.

978-0-7695-3973-7/10 $26.00 © 2010 IEEE
DOI 10.1109/ISMS.2010.42

MARTE (Modelling and analysis of Real Time Embedded Systems) [8] is an UML profile that targets real time embedded systems and tends to replace the SPT profile. It offers a facility for modelling, and analyzing real time applications. The MARTE architecture is focused on four packages: the MARTE foundations, the MARTE design model, the MARTE analysis model, and the MARTE annexes. The MARTE analysis package introduces common elements that can be used in providing input to many kinds of quantitative analysis.

Three particular types of analysis are considered, The Schedulability Analysis Modelling, the Performance Analysis Modelling and the Worst Case Execution Time Analysis Modelling.

The aim of this work is not to develop an UML profile, rather than, it aims to develop an UML tool with a minimum modelling effort for ES discrete event simulation. For this purpose, we have exploited three UML diagrams: sequence diagram to model interactions among objects, structure diagram to model the hardware platform, and the deployment diagram to model mapping.

III. OUR PROPOSED APPROACH

The proposed approach is illustrated in figure 1. Our aim is to conceive an UML2.0 based tool following the Y-chart approach that permits performances estimation of embedded systems. For this purpose, we choose the Rhapsody environment [10].

Rhapsody targets Model Driven embedded software development. It allows designers to analyze, design, implement, and test his/her system. It also allows automatic test generation, reverse engineering and roundtrip. Using Rhapsody, we can easily model, check, and simulate our application.

From UML models, an XMI (the XML Metadata Interchange language) representation is generated. One advantage of using XMI is to grant interoperability between UML tools. In order to parse the XMI file, we use the Java API called JDOM. The latter transforms the XMI representation to an abstract tree from which we extract the pertinent information to build the simulation model. The XMI file is generated inside the Rhapsody environment.

In our case, we use the sequence diagram to model the functionality of a system in term of a set of communicating objects via asynchronous messages. Each object executes a set of methods.

Here there is no explicit concurrency modelling and all messages are considered executed sequentially. In other words, our initial model is a pure sequential object paradigm. We think that such a paradigm brings many benefits. First, because an object-oriented paradigm is preferable in terms of abstraction and reuse. Secondly, we think that a sequential model facilitates the modelling task relieving the designer of the burden of concurrency modelling. Thirdly, starting from a sequential model we can then analyse the

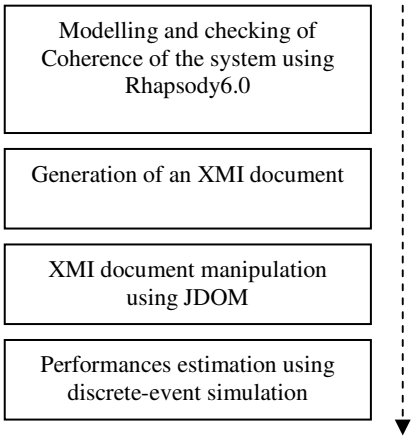

Figure 1. Our proposed approach

dependencies between messages to extract methods execute sequentially or/and concurrently. In other word, the sequential model is strongly preferred from the system designer's perspective.

In order to represent dependencies between methods and messages, we have to enrich the sequence diagram with time constraints. For each method we introduce the beginning, the execution and the end time in the worst case. For instance t, 10, t+10. Two different methods may be dependent (the output of the first method is the input of the second one), in sequence, or independent (may be executed in parallel). In the first case if the end time of a method is *t*, and the execution time equal to 10ns, the other method has a beginning time (*bt*) equal to t+10. In the second case $bt > t+10$. In the third case, if the beginning time of a method is *t*, the other method has a *bt* equal to for instance *w*. two kinds of messages are modeled: periodic and aperiodic.

The periodic messages (in the diagram they are preceded by the timer event) follow the Poisson law (the period is modelled with the timer event), and the aperiodic the normal law (the min and the max of event occurrence times are extracted from the data base). We attach to the sequence diagram a data base containing estimations on methods and messages. For instance, each method is characterized by its priority, operations number, data and code memory size (Kb) and the length of each message (in bits).

The diagram of structure models the hardware components with their characteristics and topology. We find for instance microprocessors (ex. ARM), DSP (Digital Signal Processing), FPGA (Field Programmable Gate Arrays), ASIC (Application Specific Integrated Circuit), a shared RAM, Bus and high-speed connectors. For each component, some information is provided ex. The horologe frequency, internal code and data memory size, the scheduling policy (ex. FIFO or priority-based). For buses and connectors, the transfer speed and the scheduling policy.

The mapping consists in assignment of logical components to physical ones. So methods are assigned to

processors and messages to buses or high-speed connectors. If the internal data memory of a processor is not sufficient to stock the data code of the assigned method, we use the shared memory instead. Figure 2 shows an example of a sequence diagram modeled using the Rhapsody environment.

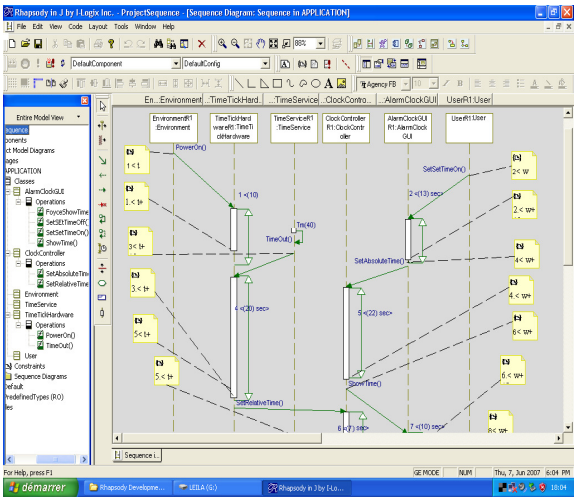

Figure 2. Sequence diagram annotated with temporal constraints

nom1	nom2	relation
PowerOn	SetSetTimeOn	independant
PowerOn	TimeOut	sequence
PowerOn	SetAbsoluteTi	independant
PowerOn	ShowTime	sequence
PowerOn	SetRelativeTin	independant
PowerOn	ForceShowTim	sequence
PowerOn	SetSEtTimeOff	independant
SetSetTimeOn	TimeOut	independant
SetSetTimeOn	SetAbsoluteTi	depend
SetSetTimeOn	ShowTime	independant
SetSetTimeOn	SetRelativeTin	sequence
SetSetTimeOn	FoyceShowTin	independant
SetSetTimeOn	SetSEtTimeOff	sequence
TimeOut	SetAbsoluteTi	independant
TimeOut	ShowTime	depend
TimeOut	SetRelativeTin	independant
TimeOut	ForceShowTim	sequence
TimeOut	SetSEtTimeOff	independant
SetAbsoluteTi	ShowTime	independant
SetAbsoluteTi	SetRelativeTin	depend
SetAbsoluteTi	ForceShowTim	independant
SetAbsoluteTi	SetSEtTimeOff	sequence
ShowTime	SetRelativeTin	independant
ShowTime	ForceShowTim	depend

Enr : 6 sur 28 Aucun filtre Rech

Figure 3. Dependencies relations between methods

Figure 3 illustrates all dependencies relations between methods. These relations are extracted automatically from the sequence diagram of figure 2.

Figure 4 shows a diagram of structure modeled via Rhapsody in which we add constraints to indicate the direction and the length in bits of each port.

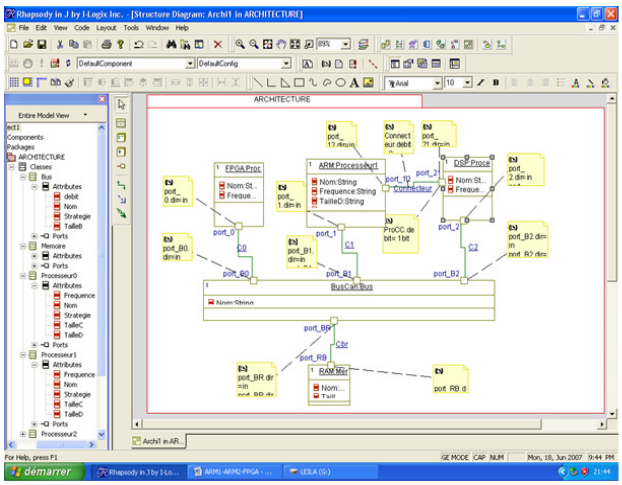

Figure 4. Modelling of Hardware platform using UML2 structure diagram

Figure 5 shows the deployment diagram, on which mapping is done. In this example, we map all transfers to a bus. Methods: PowerOn and SetSetTimeOn are mapped to FPGA, methods forceShowTime, ShowTime, and Timeout to ARM, and methods SetAbsolutTime, SetRelativeTime, and SetSetTimeOff to DSP.

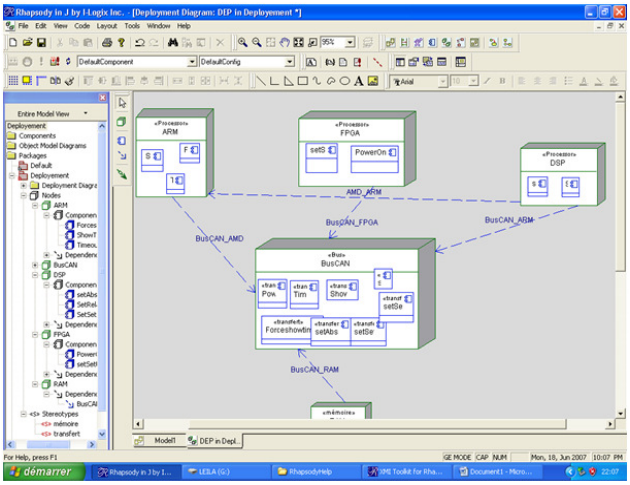

Figure 5. Modelling of mapping using UML2 deployment diagram

IV. SIMULATION

We use the discrete event simulation to estimate performances of the architectural model (the result of mapping). In our case, the servers are the processors and the buses and the clients are the methods and the messages. To each server, we assign a queue in which clients wait for execution. Events arrivals are generated in a random fashion. Events are either periodic following Poisson law or aperiodic following the Normal law.

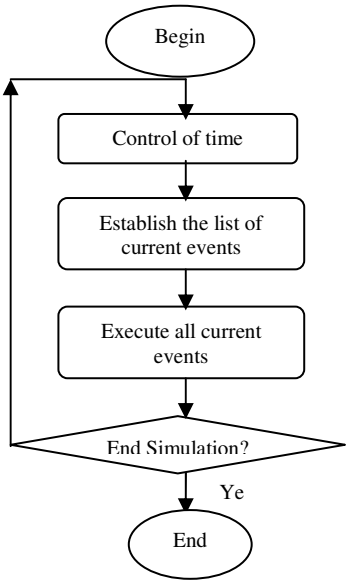

Figure 6. Steps of discrete-event simulation

On arrival of an event, the simulator tests if the event is independent of other events. In this case it will be executed directly otherwise (it depends of other events), it will be blocked in the queue until the arrival of the event on which it depend. The considered parameters are the busy time, the free time and the total time of execution. According to simulation results (see figure 7), designer can modify the mapping itself, the application or the platform parameters.

V. CONCLUSION

In this work, we presented an UML2.0 based tool that allows designers to evaluate performances of an embedded system. The tool follows the Y-chart approach, where application is modeled via sequence diagram, platform via diagram of architecture, and mapping via diagram of deployment.

Using XML, we can easily automate the passage from UML to queue models. In order to evaluate performances, we applied discrete-event simulation principles on the architectural model. As perspectives, we plan to exploit more sophisticated queue models for performances evaluation. Another aim is to improve our tool to be capable of finding automatically optimal mappings and evaluate other quality criteria as energy consumption and memory print.

REFERENCES

[1] D.D Gajski, F. vahid, S. Narayan, and J. Gong. Specification and Design of Embedded Systems. Published by Prentice Hall. Englewood, Newjersey 07632. 1994.

[2] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide* (Addison-Wesley, 1999).

[3] K. keutzer, S. Malik, R. Newton, j. Rabaey, and A. Sangiovanni-Vincentelli. System level design: orthgonalization of concerns and Platform-Based Design. In IEEE transactions on computer-aided design of circuits and systems, Vol. 19. , No. 12, December 2000.

[4] B. Kienhuis, Ed F. Deprettere, P.V. Wolf, and K. Vissers. A Methodology to design Programmable Embedded Systems. The Y-chart approach. In LNCS series vol. 2268, page 18-37 by Springer Verlag © 2001.

[5] G. Martin. UML for embedded systems specification and design: motivation and overview. Cadence design systems, 2002.

[6] L. Lavagno, G. Martin, and B. Selic. UML for Real: Design of Embedded Real Time Systems. Kluwer Academic Publishers. Print ISBN 1-4020-7501-4. 2003.

[7] F. Boutekkouk and M. Benmohammed. Analyse dirigée par les scénarios de l'ordonnaçabilité d'une application embarquée temps réel. Dans CGE'05. Ecole militaire polytechnique, Bordj El Bahri, Alger, 16-17 Avril 2007.

[8] OMG. UML Profile for MARTE, Beta 1. *OMG Adopted Specification, ptc/07-08-04,* August 2007.

[9] F. Boutekkouk, M. Benmohammed, S. Bilavarn, M. Auguin, UML2.0 profiles for Embedded Systems and Systems on a Chip (SOCs). In *JOT (Journal of Object Technology)*, January 2009.

[10] www.ilogix.com.

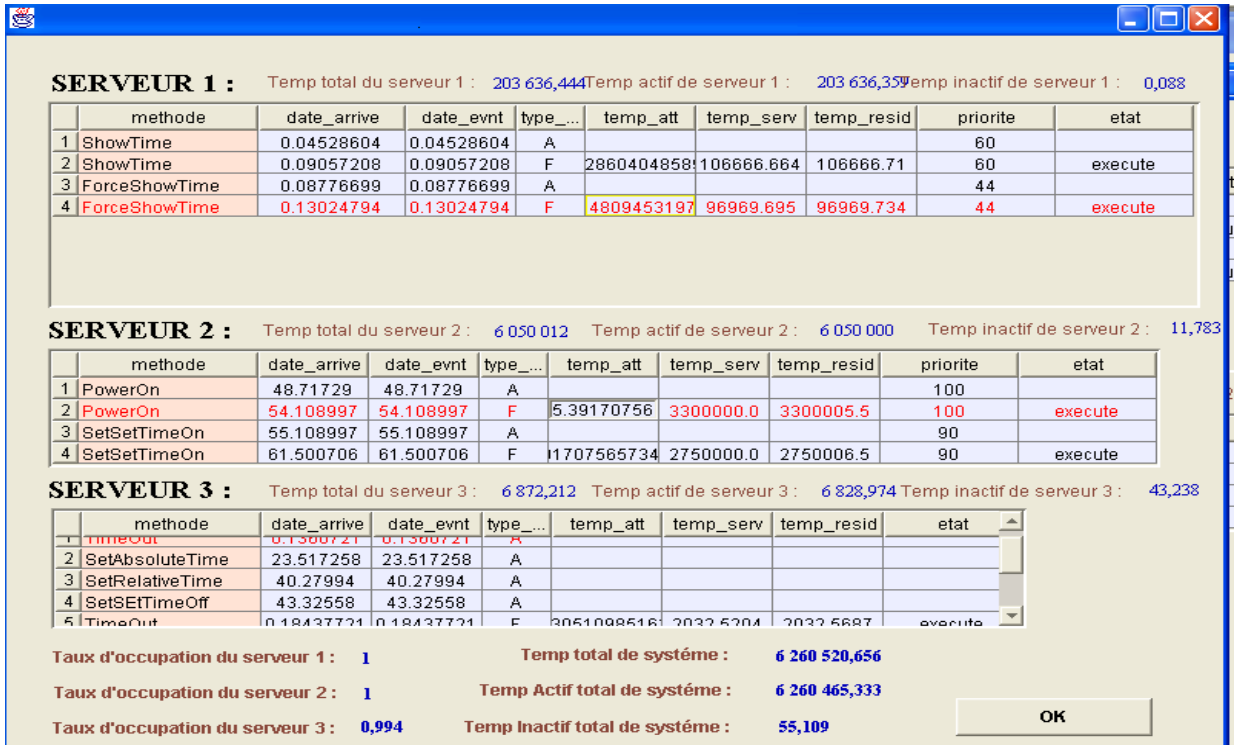

Figure 7. Results of simulation