Towards Model-Driven Engineering for Mixed-Criticality Systems: MultiPARTES Approach

Alejandro Alonso*, Christophe Jouvray†, Salvador Trujillo‡, Miguel A. de Miguel*, Cyril Grepet †, José Simó§
*Universidad Politécnica de Madrid, Spain. Email:aalonso, mmiguel@dit.upm.es
†TRIALOG, France. Email: christophe.jouvray, cyril.grepet@trialog.com
‡Ikerlan-IK4, Spain. Email: STrujillo@ikerlan.es
§ Universidad Politécnica de Valencia, Spain. Email: jsimo@disca.upv.es

***Abstract*—Mixed criticality systems emerges as a suitable solution for dealing with the complexity, performance and costs of future embedded and dependable systems. However, this paradigm adds additional complexity to their development. This paper proposes an approach for dealing with this scenario that relies on hardware virtualization and Model-Driven Engineering (MDE). Hardware virtualization ensures isolation between subsystems with different criticality levels. MDE is intended to bridge the gap between design issues and partitioning concerns. MDE tooling will enhance the functional models by annotating partitioning and extra-functional properties. System partitioning and subsystems allocation will be generated with a high degree of automation. System configuration will be validated for ensuring that the resources assigned to a partition are sufficient for executing the allocated software components and that time requirements are met.**

Keywords: Mixed criticality systems, model-driven engineering, virtualization, embedded systems

I. Introduction

Modern embedded applications typically integrate a multitude of functionalities with potentially different criticality levels into a single system. Without appropriate preconditions, the integration of mixed-criticality subsystems can lead to a significant and potentially unacceptable increase of certification efforts. One approach to avoid the increased validation and certification effort is to incorporate mechanisms that establish multiple partitions with strict temporal and spatial separation between them. In this approach, subsystems with different levels of criticality can be placed in different partitions and can be verified and validated in isolation.

Model Driven Engineering (MDE) is a particularly promising approach in this setting, as it facilitates to bridge the gap between design issues and partitioning concerns. MDE is changing the way systems are developed nowadays, reducing development time significantly. In general, modeling approaches have shown their benefits when applied to embedded systems [11]. These benefits have been achieved by fostering reuse with an intensive use of abstractions, or automating the generation of boiler-plate code. However, MDE has not been applied to systems with mixed levels of criticality.

The originality of this paper lies in the propola of a MDE approach for the engineering of mixed-criticality systems intended for heterogeneous multi-core platforms based on hypervisor techniques. MultiPARTES [12] is an EU funded FP7 project that aims at supporting the development of mixed-criticality embedded applications on multi-core platforms, based on a hypervisor and MDE tools. MultiPARTES will offer a rapid and cost-effective approach for the engineering of dependable real-time embedded systems integrating critical and non-critical applications sharing system resources.

The ultimate goal of MultiPARTES is to devise a comprehensive engineering methodology to take full advantage of partitioning of multi-core systems, thereby speeding up development and production of new highly dependable and secure applications. The results will be validated in several application sectors: offshore wind power, industrial control, video surveillance, and space.

II. Motivation and background

The paper presents a new approach in order to manage mixed-criticality application design. This work relies on a dedicated platform based on hypervisor. This section first presents the architecture and then the interest of hypervisors.

A. Space and Time Separation of Mixed-Criticality Applications

For reducing costs, mixing different applications in an unique processing unit is advantageous. Unfortunately, it seems difficult to associate several application with different level of criticality. A solution consists in using an architecture providing space and time separation between applications.

MultiPARTES proposes to use the architecture depicted in Figure 1. The isolation layer constitutes the Trusted Computing Base (TCB).

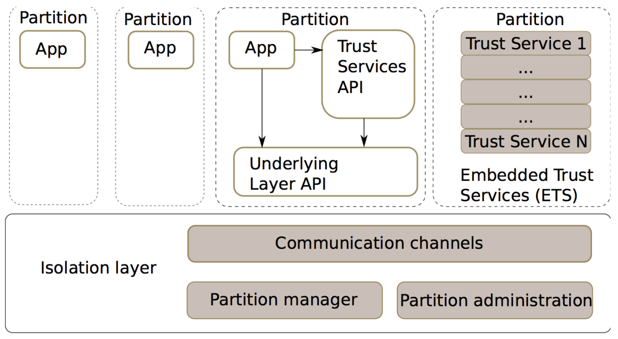

Fig. 1. Abstract Architecture Supporting Mixed-Criticality Applications

The isolation layer provides the means to perform software partitioning and abstracts the underlying resources (e.g. hardware resources) for the software stack. This component is considered trusted, as it is the only one that sets up the isolation mechanism. The isolation layer includes three parts:

1) The partition manager or kernel, or the part that implements the isolation mechanisms;
2) The communication channels, which are the channels through which partitions can communicate, under the control of the TCB that enforces the security and safety policies;
3) The partition administration, which provides an interface to administrate the kernel, in particular the partitions and the communication channels between them. It also provides the various system services needed to abstract the lower layers such as the device drivers.

The combination of the isolation layer with all lower implementation layers (i.e., hardware) makes up the trusted computing base (TCB).

A partition includes isolated applications. This application accesses services of the underlying layers through appropriate APIs, including a specific API for the embedded trust services.

The rational of this architecture can be summarized by 3 main principles:

- ***Principle 1***: Small TCB. The Trusted Computing Based (TCB) has to provide the strict minimal features in order to provide an isolation mechanism. The lower the level of complexity of the TCB, the higher the level of trust.
- ***Principle 2***: Partitioning. Applications are isolated in a partition. Due to the isolation, a fault is confined in a partition and cannot be propagated to another partition.
- ***Principle 3***: ETS Extension. A dedicated partition hosts the Embedded Trust Services (ETS) which provides powerful security and dependability services. The ETS can be called from an application via a dedicated API.

In order to implement the isolation layer of the TCB, different solutions are possible such as hypervisors, which is used in this work and is presented in the next section.

B. Hypervisors

Virtual machine technology is often considered as the most secure and efficient way to build partitioned systems. A virtual machine (VM) is a software implementation of a machine (computer) that executes programs like a real machine. Although the basic idea of virtualizing [7] is widely understood: any way to recreate an execution environment, which is not the original (native) one; there are substantial differences between the different technological approaches used to achieve this goal [17].

Hypervisor (also known as virtual machine monitor VMM [5]) is is a thin layer of software that virtualizes the critical hardware devices to create several isolated execution environments also known as partitions. Two types of hypervisors can be found:

- Type 1 are hypervisors running directly on the native hardware (also bare-metal hypervisors)
- Type 2 of hypervisors are executed on top of an operating system.

The key difference between hypervisor technology and other kind of virtualisation (such as Java virtual machine or software emulation) is the performance. In bare-metal hypervisors the overhead can be very low maintaining the throughput of the virtual machines very close to the native hardware.

XtratuM [9], [3] is a bare-metal hypervisor that has been designed specifically for secure and critical real-time embedded systems following a set of requirements for secure space applications and a set of services to build applications based on the ARINC-653 standard [1].

XtratuM does not define a new abstract virtual machine but tries to reuse and adapt to the underlying hardware as much as possible to reduce the virtualization overhead. In other words, the virtual machine will be close to the native hardware in order to directly use the native hardware as much as possible without jeopardizing the temporal and spatial isolation.

The basic properties of XtratuM are related to the temporal and spatial isolation of the partitions.

- Strong spatial isolation: Hypervisor has to be executed in privilege processor mode, whereas partitions are executed in user processor mode. Partitions are allocated in independent physical memory addresses. Partitions can not access to other partition memory addresses.
- Strong temporal isolation: Hypervisor enforces the temporal isolation by using the appropriated scheduling policies to execute partitions.
- Resource allocation: Fine grain hardware and software resource allocation is specified in a system configuration file.
- Deterministic services: All services (hypercalls) provided by the hypervisor are deterministic to guarantee the real time operations of partitions.
- Fault management model: Faults are detected and handled by the hypervisor.

Relying on an hypervisor-based architecture, MultiPARTES proposed a new approach for designing mixed-criticality applications. This approach is presented in the next section.

III. MultiPARTES Development Process

A MultiPARTES system is formed by a set of applications, which are composed by a number of interacting software components or artifacts. Software components of an application may run in the same or different partition. They can also be allocated on different cores.

Software components can have different levels of criticality. An important requirement is to ensure that each component runs according to their criticality requirements, without interference from other applications, running on the same platform and with lower criticality level.

MDE tools will be used for describing the system. In particular, at least the following descriptions will be required:

- A high-level system model, that is composed by a set of software components and their interconnections. Software

components are described by a functional model, that can be annotated with non-functional information. In this work, non-functional properties that have a direct influence on system partitioning are of special relevance.
- A high-level model of the hardware resources. It includes hardware devices and the available resources quantities.
- Available resources management models. The management of resources is important for deriving system behavior, such as time response or resource availability. In particular, the scheduling policy of the operating system running on a partition or the communication protocols are needed for deriving system properties.

Given these inputs, a partitioning tool will be in charge of generating a system partitioning, including the allocation of resources and software components to partitions. The feasibility of this partitioning should be validated against a number of constraints, mainly derived from the non-functional requirements. Some relevant constraints, in the expected application domains of the MultiPARTES tools, are:

- Criticality requirements: the assignment of software components to partitions and the relations between components should avoid dependencies inversion. This implies that critical components may use, but do not depend on less critical components [18].
- Time requirements: the partitioning configuration and the software component configuration should be such that time requirements are met. A model for response-time analysis should be derived from the system description, which will be the input to an analysis tool, for checking their fulfillment.
- Resources availability requirements: The software components may require a certain amount of resources for fulfilling non-functional requirements. If this information is included as annotations, it would be possible to check whether there are enough resources on a given partition for running properly these components.
- Device usage requirements: In embedded systems it is common to use special devices, such as sensors and actuators. It has to be considered whether a certain software component requires a given device, in order to assign it to a processor that can access it.

In the case of time and resource usage requirements, it is possible to use response time analysis tools. Currently, there are some analysis tools available, such as [14] that provides response time analysis based on fundamental scheduling analysis models. However, they must be improved in order to handle the type of applications and execution platforms targeted in MultiPARTES.

A different analysis tool will be used for ensuring the validity of a partitioning with respect to safety constraints. It will check whether there are no dependency inversion situations. If it is not feasible to generate a partitioning without this problem, the faulty relations will be notified to the designer for fixing them.

Once a system partitioning has been validated, a number of outcomes should be generated, such as:

- A partitioning of the system, including the final configuration of each of the partitions.
- An assignment of software components to partitions.
- Skeletons of code for the software components. They should be adapted to the selected programming language or operating system, and shall include configuration parameters, such as priorities for competing for resources.

IV. MDE for Designing Mixed-Criticality Applications

Developing critical systems is complex and need a dedicated methodology. The methodology must follow the overall process and propose different tools in order to help the designers and to avoid errors.

A. Overview of the Approach

MultiPARTES methodology is based on metamodeling and is summarized by Figure 2. The goal of a metamodel is to have a common understanding by defining concepts and their relationships. For the concern addressed by this paper, our approach is based on several metamodels described in the followings:

- *Execution Platform Metamodel*. This metamodel is dedicated to concepts related to platforms such as multi-core processors, memory, bus, etc. The state of the art already proposes solution such as MARTE [21].
- *Logical Platform Metamodel or Execution Environment Metamodel*. Compared to the previous metamodel, this one is focused on abstract concepts (i.e., partition). This metamodel is a contribution of MultiPARTES.
- *Application Metamodel*. This metamodel focuses on concepts related to application design.

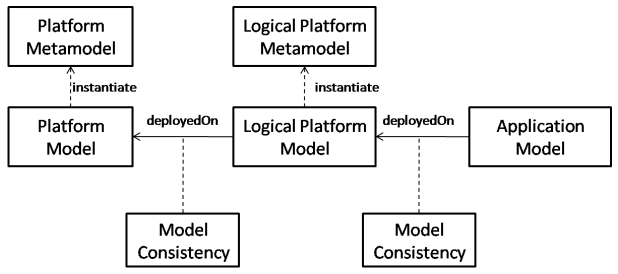

Fig. 2. Overview of the MultiPARTES Methodology

A metamodel can be considered as a new language. It is possible to develop some models conformed to a metamodel (i.e., represented by the link "instantiate" in the Figure 2). The metamodel instantiation can be processed over different ways: (i) Domain Specific Languages (DSL), (ii) Domain Specific Model Languages and (iii) UML Profiles.

The different applications are modeled (e.g., in UML or SysML). Consistency checking is needed at different levels are represented in Figure 2 by association classes. Indeed, the application will be restricted by requirements related to the platform. The logical and real platform must support the resource application needs.

In MultiPARTES, annotation solution based on MARTE is used. This means that a UML model is used and enriched in order to provide all needed information for checking the models consistency. In particular, properties at application level must be supported by the execution environment.

Figure 3 shows the Execution Environment metamodel. Its goal is to define all concepts needed for characterizing a logical platform, such as a partition. In Figure 3, the ETS corresponds to a dedicated partition. This partition provides different services to applications of any other partition.

A partition is characterized by different properties such as physical memory accesses, processor which executes the partition, and communication ports. This information is mandatory in order to configure the hypervisor. Moreover, it is needed to check that application needs (i.e., memory, communication) are fulfilled by the partition.

V. Non-functional Annotations for Partitioning

The purpose of this section is to identify the annotations to be included in functional models for driving the partitioning process. In particular, these annotations are used for deciding on how many partitions must be created, on their configuration, on the software artifacts that will be included on each of them, an on their validation. One aim of this work is to generate and validate system partitioning in an automatic way. These annotations will also be used by analysis tools to ensure that the generated partitioning meets non-functional constraints.

The most relevant types of constraints were identified on section III and are dealt with in this section.

Safety requirements

The aim of safety annotations are to classify each component according to its criticality in a functional system design. They will be used in MultiPARTES to identify components safety level, in order to group in the same partition components with the same level, and to identify potential dependency inversion situations.

Annotations will mainly be based on the criticality level or software integrity level of standards related with the specific application domains. The European Space Agency has produced a number of standards for safety. It identifies four criticality levels [4]. Standards for different application domains use to identify similar criticality levels.

Criticality constraints will be considered during the partition process. Components with the same criticality level and related, from a functional point of view, will be located in the same virtual machine. In addition, partitions with high criticality level will be kept as simple as possible, for facilitating their validation and certification.

Time requirements

The structure of a real-time safety application can follow the linear model [6] [13]. They are structured as a number of end-to-end flows (using the MARTE concepts). Each flow is composed of a set of ordered steps and is characterized by a sporadic or periodic activation pattern. A step can be the execution of a code segment or a message sent by the communication media, if segments of code are in different partitions or cores. Every step is released by a previous one, except for the first which is activated according to the flow pattern.

Time requirements are usually defined as deadlines on the maximum elapsed time between the activation of one of such flows and the completion of its execution. In some cases, partial deadlines can be defined with respect to the operation of a subset of connected steps.

In order to check the fulfillment of these deadlines, it is needed to describe the time behavior of the application, which depends on:

- *Activation pattern:* As mentioned above, two main patterns are considered: periodic and sporadic. In both cases, the flows are activated a potentially infinite number of times. In the periodic pattern, the flow is activated strictly periodically, with a fixed time interval. In the sporadic pattern, the flow is activated in response to an event which may occur once a minimum inter-arrival time has elapsed since the occurrence of the previous event activating the same flow. The annotations to be provided shall specify these intervals for each flow.
- *Resource usage of the steps:* In order to check whether a given flow meets its time requirements, it is needed to know the resources required for accomplishing their job. On components with high-criticality level, it is provided its worst case execution time. This parameter and the activation interval define the percentage of processor that the component requires. In a similar way, it is possible to bound the required bandwidth for the communication steps in the flow. The memory required is an important parameter for checking resource availability on a partition, although has no influence on the time behavior analysis.
- *Use of shared resources:* It is important to analyze the interferences between two flows sharing the same resources. Special contention policies are used for ensuring a deterministic and desired behavior. These resources can be shared memory or some special device that must be used with mutual exclusion.

MARTE [21]will be used as the basis for describing these annotations. Section VII includes an overview of this standard.

Resource usage requirements

There are applications with non-critical time requirements or with quality requirements, where occasional failing some of them is acceptable. In this case, it is not necessary to allocate resources for ensuring a proper execution in all execution scenarios. In turn, it is sufficient to assign resources for working as desired most of the time, while keeping the less optimal behavior situations under a given threshold. These software components are annotated with an estimation of the resources required. Then, it is possible to check whether a

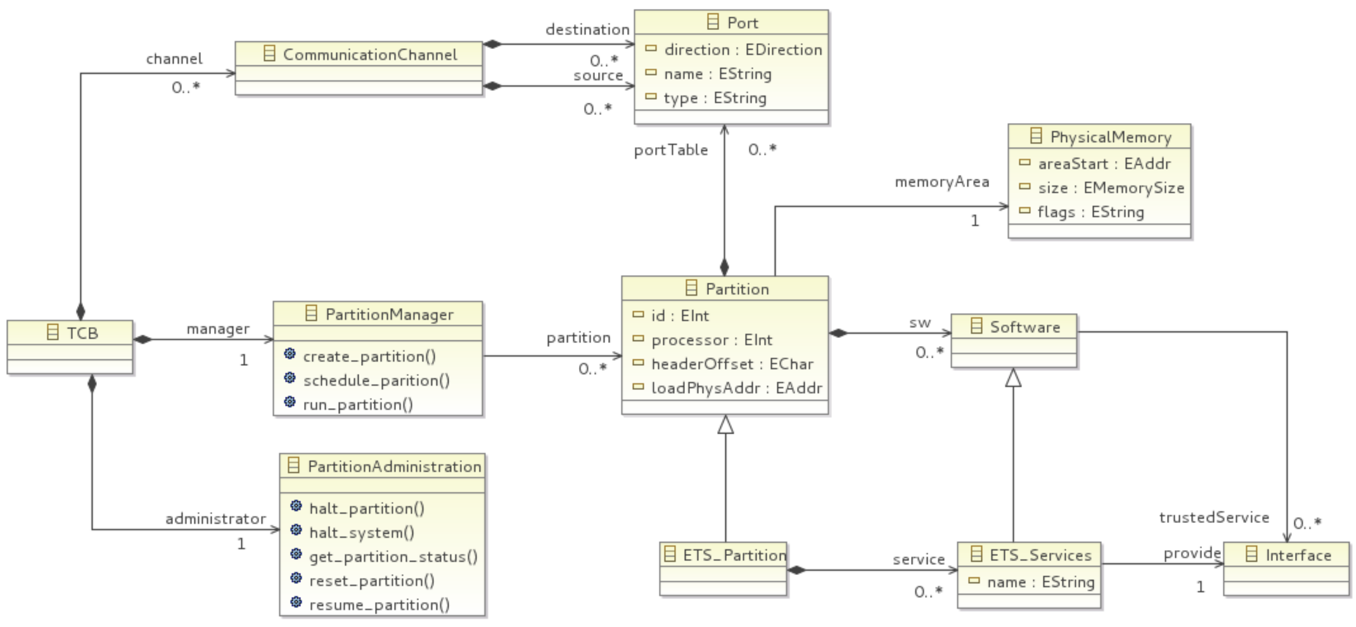

Fig. 3. Overview of the Execution Environment Metamodel

given partition has enough resources assigned to satisfy those required by the allocated artifacts. A certain CPU percentage, an amount of memory, or a given bandwidth are examples of such resource requirements. The tools and notations to be used for the requirements in the previous section, are also appropriate for describing resource usage parameters.

Device usage requirements

In an embedded system it is common to use special devices, such as sensors or actuators. According to the hardware architecture of a system, it may be more simple to access to one of such devices from a particular core or processor. It is convenient to indicate for a software component which special devices do it need, in order to allocate it on a partition running on a proper core or processor.

VI. Case Studies

This section summarizes the selected industrial use cases to demonstrate the benefits of the MultiPARTES approach. Five use cases from different sectors have been selected by the industrial partners, namely, space, visual surveillance, automotive, railway and offshore wind power [20].

MDE has been used mostly to generate software or, to some extent, to generate systems artifacts. This work uncovers a big potential by applying MDE beyond conventional software engineering approaches. Mixed-criticality systems encompass further hardware related artifacts and lower-level configuration/specification files. For instance, MultiPARTES will generate platform and execution environment models for those application models that are to be executed on top of the hardware platform. This provides a fresher perspective on the MDE field. Similarly, this constitutes a significant automation of repetitive configuration work where checking consistencies is time-consuming. In general, any approach with the potential to automate and partially reduce the manual activities may foster productivity.

Each use case description has special focus on particular elements of the MultiPARTES approach. The overall goal is to improve the development of currently existing and future products by means of the mixed-criticality modeling approach. The ultimate goal is for the project to focus on providing a practical solution for industrial practitioners from assorted sectors.

The common benefits seek in all the use cases are related to reduction of engineering effort, reduction of time to integrate a third-party application, reduction of time for certification, and so on. A higher performance is achieved at the cost of increasing the consumption of energy, volume, and weight. This is an interesting trade-off where throughput is improved as a result. Time-to-market pressures are fundamental in offshore wind power, safety and certification are key in railway, while quickly integrating third-party applications into the platform is a market advantage in visual surveillance. MultiPARTES will provide benefits in each of these fields. The modeling approach described in this paper will play a pivotal role in achieving those benefits.

VII. Related Work

MultiPARTES is addressing open technological issues not addressed yet in the specific MDE domains and techniques specializations (e.g. real-time MDE, high integrity MDE). These issues include: i) virtualization effects in deployment specifications, ii) combination of different types of critical requirements (e.g. safety critical and business critical), and iii) multicore technologies.

Current MDE technologies for high integrity applications (e.g. MARTE [21], UML Profile for QoS [22]) cannot represent important topics in the analysis and development of critical systems, such as the effect of virtual cores and caches in response time analysis, and the deployment into virtual partitions.

MultiPARTES issues are not well handled yet in analysis tools and methods for the evaluation of high integrity applications. Analysis methods such as scheduling analysis and fault analysis does not take into account these issues and the analysis tools require important improvements.

The code generation in MDD technologies for high integrity applications does address important issues such as the deployment of software artifacts into partitions and cores, and the integration of analysis methods and code generation is very limited.

This section introduces current technologies for the specification, analysis and code generation of high integrity applications, and the most important issues to be addressed in these technologies for their application in MultiPARTES software MDD.

A. *Model-Based Specification of High Integrity Applications*

MARTE is the OMG standard for the specification of real-time systems. MARTE addresses different modeling concepts in a number of profiles. In the context of this work, the most important are: the analysis of UML models (profiles GQAM, GR, SAM, and PAM), and the design of real-time software, taking into account hardware concepts (profiles GRM, DRM, GCM, and HLAM).

MARTE requires important improvements to overcome some of its limitations, including

- *Profiles redundancy*. In MARTE, different profiles represent the some concepts/values several times. For example, the period of a flow is specified for work load events in analysis, schedulable resources in design, and real-time service specification. This redundancy makes complex to maintain the models intended for multiple purposes (analysis and software development).
- *Independent analysis model*. MARTE analysis models are practically independent of UML models (GQAM, PAM and SAM extension applications can be represented, practically, as independent abstract syntax trees). This approach can create inconsistencies between development and analysis concepts.
- *Semantic of MARTE languages*. MARTE profiles include some natural language constraints, which are imprecise and limited. Important concepts such as precedence or incompatibility of redundant concepts are not specified. For example, it is possible to associate different deadlines to the same step, without fixing their precedence.
- *Complex values specification*. MARTE extensions include complex values specifications (e.g. specification of release patterns), and MARTE reuses other languages (e.g. VSL) to improve its usability. However, the complexity of these extensions and MARTE difficulties their applicability.

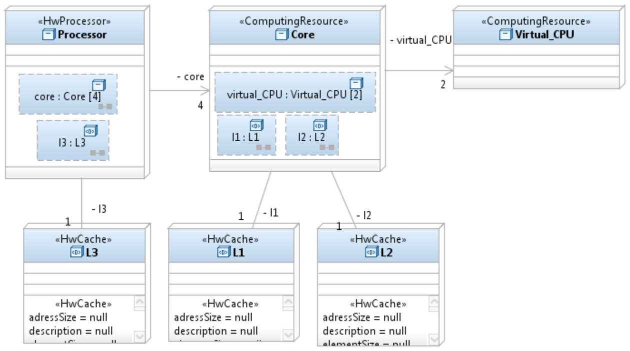

Fig. 4. MARTE representation of the Intel Core i7 2820QM structure

Code and analysis models generators that reuse MARTE must introduce additional constraints or language limitations to be more precise and to avoid important problems of its applicability.

Some important concepts in MultiPARTES that are not fully addressed in MARTE include:

1) Specification of multicore hardware and software concepts (e.g. virtual CPU, CPU core, cache levels, CPU affinities).
2) Specification of partitions concepts.
3) Memory dependencies of cores. The memory models of multicore architectures have an important effect in software performance. The analysis of software designs must take into account these hardware properties.
4) Multilevel caches of multicore processors can produce unpredictability.

Figure 4 is an example of application of MARTE for the representation of multicore Intel Core i7 2820QM structure in a deployment model. However, some concepts cannot be represented, such as the difference between core and virtual processor, and dependencies and policies of cache memories.

B. *Improvement of analysis tools*

There are response analysis tools, such as MAST [14] that can be used for validating a system partitioning with respect to time and resources usage requirements. This tool allows the analysis of a large variety of real-time systems and scheduling policies. The support to multi-cores is limited, although some support is provided for multi-processor systems.

The MAST analysis tool needs some improvements for meeting the characteristics of the domains targeted by MultiPARTES. The support for virtualized systems is limited. The tool includes means for hierarchical scheduling, but additional scheduling policies must be included. In addition, other effects of virtualization and partitioning on multi-cores shall be integrated in the analysis.

The effect of caches in the response time of an application is important. General purpose processors (e.g. Intel Core) include several levels of cache and cores share caches at some levels.

There are some works [19], [16] that analyse cache effects. However, further work is needed for integrating in the response time analysis these hardware components and architectures.

Mixed criticality systems require specific analysis approaches for the isolation of software artifacts with different levels of criticality. Deployment models must be improved to detect and handle these kinds of issues.

C. Improvements of development tools

MDE allows for reducing software development costs cost of software developments by increasing the levels of abstraction in software development processes and employing code generation techniques to reuse software design patterns. Code generation tools for high criticality systems relies on specific language profiles (e.g. Ada RAVENSCAR profile [2] and Safety-Critical Java [8]). These profiles introduces restrictions in the programming languages and execution platforms, in order to guarantee full system predictability. General and specialized code generation approaches [15] [10] for high integrity application must force these restriction in the software patterns in the generators, and in the logical code that provide the application modeler.

Another important concept is the customization of deployment generators for the integration of virtualization and partitioning configurations. Deployment configuration must conform with analysis models, multicore platform structure, and restrictions of critically for the different partitions.

VIII. Conclusion

This paper introduced an approach for the engineering of mixed criticality systems based on heterogeneous multicore and hypervisor technology. The necessity for this novel approach appears in different industrial sectors. This case is expected to become a typical scenario in the future of embedded systems engineering. Multicore open source virtualization appears as a potential candidate solution for addressing future challenges, namely, reducing costs, power consumption, volume, and time to market, in an effective way.

The major benefits of the MultiPARTES modeling approach will come from a reduction of the time to market and the application of modeling beyond conventional boundaries.

Currently it is being carried out the definition of the hardware platform and the multicore hypervisor that will provide the basis described in this paper. The Metamodels and the tools that will empower the approach and enable its adoption in the described cases are currently under development.

Acknowledgements

The work presented in this paper is partially funded by FP7 STREP MultiPARTES project, no 287702 (www.multipartes.eu). The MultiPARTES consortium comprises: Ikerlan-IK4 (E), Universitat Politècnica de València (E), Technische Universitt Wien (A), Universidad Politécnica de Madrid (E), TRIALOG (F), Fent Innovative Software Solutions (E), TELETEL (Gr), Visual Tools (E), ALSTOM Wind (E). The work at Universidad Politécnica de Madrid has also been funded by the Spanish *Ministerio de Educación Ciencia*, project HI-PARTES (High Integrity Partitioned embedded systems), TIN2011- 28567-C03-01 in the *Plan Nacional de I+D+i*.

The authors wish to acknowledge the help and contribution of other members of the MultiPARTES consortium.

References

[1] Airlines Electronic Engineering Committee: Application Software Standard Interface (ARINC-653), 2551 Riva Road, Annapolis, Maryland 21401-7435 (1996)

[2] Burns, A., Dobbing, B. and Vardanega, T.: Guide for the use of the Ada RAVENSCAR profile in high integrity systems, Ada Lett. XXIV(2): 174, (2004).

[3] A. Crespo, I. Ripoll, M. Masmano, S. Peiro: Partitioned Embedded Architecture Based on Hypervisor: The XtratuM Approach, EDCC, 67–72 (2010)

[4] European Space Agency, ECSS-Q-ST-80CSpaceProductAssuranceSoftwareProductAssurance (2009).

[5] R.P. Goldberg:Survey of Virtual Machine Research, IEEE Computer Magazine, vol. 6, no. 6, 34–45, (1974)

[6] Gutiérrez, J. J., Palencia, J. C. and González Harbour, M.: Schedulability analysis of distributed hard real-time systems with multiple-event synchronization, Proc. 12th Euromicro Conference on Real-Time Systems, IEEE CS Press, pp. 1524 (2000).

[7] IBM Corporation: IBM Systems Virtualization. Version 2 Release 1 (2005) http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicay/-eicay.pdf

[8] JSR 302: Safety Critical JavaTM Technology, http://jcp.org/en/jsr/detail?id=302

[9] M. Masmano, I. Ripoll, A. Crespo, J.J. Metge, P. Arberet: XTRATUM: An open source hypervisor for TSP embedded systems in aerospace, DASIA 2009. DAta Systems In Aerospace, (2009).

[10] Méry, D. , Singh, N.KA generic framework: From modeling to code. Innovations in Systems and Software Engineering Volume 7, Issue 4, Pages 227-235 (2011).

[11] P. Mohagheghi, V. Dehlen: Where Is the Proof? - A Review of Experiences from Applying MDE in Industry, 4th European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA 2008)

[12] MultiPARTES: Multi-cores Partitioning for Trusted Embedded Systems, EU FP7 STREP project, no 287702, http://www.multipartes.eu

[13] Palencia, J. C., Gutiérrez, J. J., González Harbour, M.: On the schedulability analysis for distributed hard real-time systems, Proc 9th Euromicro Workshop on Real-Time Systems, IEEE CS Press, pp. 136143, (1997).

[14] Pasaje, J., Harbour, M. and Drake, J.: MAST real-time view: a graphic UML tool for modeling object-oriented real-time systems, 22nd IEEE Real-Time Systems Symposium pp. 245256 (2001).

[15] R. Quadri, I.a , Yu, H.b , Gamati, A.a , Rutten, E.c , Meftali, S.a , Dekeyser, J.-L.: Targeting reconfigurable FPGA based SoCs using the UML MARTE profile: From high abstraction levels to code generation International Journal of Embedded Systems Volume 4, Issue 3-4, pages 204-224 (2010)

[16] J. Reineke, D. Grund, C. Berg, R. Wilhelm Timing Predictability of Cache Replacement Policies, Real-Time Systems, 37 (2), (2007)

[17] SCOPE Promoting Open Carrier Grade Base Platforms: Virtualization: State of the Art, http://www.scope-alliance.org, (2008).

[18] Sha, L.: Resilient Mixed-Criticality Systems, The Journal of Defense Software Engineering, vol. 22, no. 4, 9–14 (2009)

[19] Jan Staschulat, Simon Schliecker, Rolf Ernst, Scheduling Analysis of Real-Time Systems with Precise Modeling of Cache Related Preemption Delay,Proceedings of the 17th Euromicro Conference on Real-Time Systems (2005).

[20] S. Trujillo, J. M. Garate, R. E. Lopez-Herrejon, X. Mendialdua, A. Rosado, A. Egyed, C. W. Krueger, J. de Sosa: Coping with Variability in Model-Based Systems Engineering: An Experience in Green Energy, IN ECMFA (2010)

[21] UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, Version 1.1, OMG ptc/2010-08-32

[22] UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms, Version 1.1, OMG formal/2008-04-05