Toolset for Mixed-Criticality Partitioned Systems: Partitioning Algorithm and Extensibility Support

Alejandro Alonso, Emilio Salazar
Dept. de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Spain,
Email: {aalonso, esalazar}@dit.upm.es

Abstract

The development of mixed-criticality virtualized multi-core systems poses new challenges that are being subject of active research work. There is an additional complexity: it is now required to identify a set of partitions, and allocate applications to partitions. In this job, a number of issues have to be considered, such as the criticality level of the application, security and dependability requirements, operating system used by the application, time requirements granularity, specific hardware needs, etc. MultiPARTES [6] toolset relies on Model Driven Engineering (MDE) [12], which is a suitable approach in this setting. In this paper, it is described the support provided for automatic system partitioning generation and toolset extensibility.

Keywords: Real-time systems, Partitioned Systems, Mixed Criticality, Model Driven Engineering.

1 Introduction

The increasing power of processing hardware makes it possible to integrate system functionality in just one processor, instead of using several ones. Although this has a number of advantages, it presents a major problem when developing complex embedded systems. It is common that these systems include applications with different criticality level. This type of systems is called mixed-criticality. This approach presents new challenges, as it is necessary to certify the whole system, even though there are parts that are no critical.

A suitable approach is based on system virtualization. A virtualization kernel or hypervisor allows the creation of partitions that are isolated. Applications with different criticality level are executed in different partitions in a safe way.

MultiPARTES is a FP7 project aimed at developing tools and solutions for building trusted embedded systems with mixed criticality components on multicore platforms. The approach is based on an innovative open-source multicore-platform virtualization layer based on the XtratuM hypervisor. A software development methodology and an associated toolset will be provided, in order to enable trusted real-time embedded systems to be built as partitioned applications, in a timely and cost-effective way.

XtratuM [10] [5] is based on para-virtualization, which means that a given operating system has to be adapted for being able to run on top of the hypervisor. This improves system performance and predictability, making it suitable for real-time systems. XtratuM has been designed for providing spatial and space isolation. Partitions scheduling is based on a cyclic policy, that it is statically generated, compliant with ARINC-653 [2]. It precisely states when each partition has to be executed. XtratuM also supports multi-core processors.

In this paper, some aspects of the MultiPARTES toolset [11] [1] are presented. Its main goal is to support the development of mixed-criticality multi-core partitioned systems. The toolset integrates a number of tools for supporting activities such as system modelling, system partitioning, validation, and system building.

2 Toolset requirements

The development of the toolset has been driven by the requirements specification in [7]. It was mainly defined by the consortium, which is composed by academia, research institutes, and industrial partners, from the automotive, railway, space, video surveillance, and wind power domains. This specification has been refined with the comments from experts in the project Advisory Board. The most relevant requirements are summarized below.

- *Development of mixed-criticality systems:* The toolset is aimed at supporting the development of mixed-criticality systems. This implies that the concept of criticality is central in the whole development process. The criticality level of each application has to be stated.
- *System model:* The toolset has to provide means for modeling the whole system, which includes the applications, platform, and any other information that the developer has to provide for performing the requested functionality.
- *Support for non-functional requirements:* Non-functional requirements are of great importance when dealing with embedded systems. Time, safety, and security, are of non-functional requirements that will be supported. The toolset has to provide means for specifying them, and validating their fulfillment.
- *Support for partitioned systems:* System partitioning is a fundamental activity on the target type of systems. However, there is little support in similar development tools. This toolset should generate system partitioning that has to be compliant with the system models and non-functional requirements.

- *Support for multi-core architectures:* The execution platform can be multi-core, as it is commonplace in current industrial systems. The toolset shall support modeling multi-core systems and assigning partitions to cores.

- *Validation and consistency:* The toolset performs a number of models transformations, and artefacts generation. An aim of this work is to ensure that these outcomes are valid with respect to the system requirements. These objectives are considered in the implementation of the transformers. In addition, the toolset allows the integration of validation tools for performing checks when required.

- *Support for legacy systems:* It is common in industry to have applications that have been developed in the past, perhaps with different methods and tools. The toolset will provide means for allowing the integration of this type of applications in the development flow.

- *Support system deployment:* Deployment is the last step required before running the system. When dealing with partitioned embedded systems, this implies the generation of a bootable software image that includes the hypervisor, the partitions, and their operating system and applications. The toolset supports system deployment by generating mechanisms for the automatic building of the system. System deployment also requires the configuration of XtratuM.

3 Toolset architecture

The main components of the toolset and data flows are depicted in figure 1. Their basic role is:

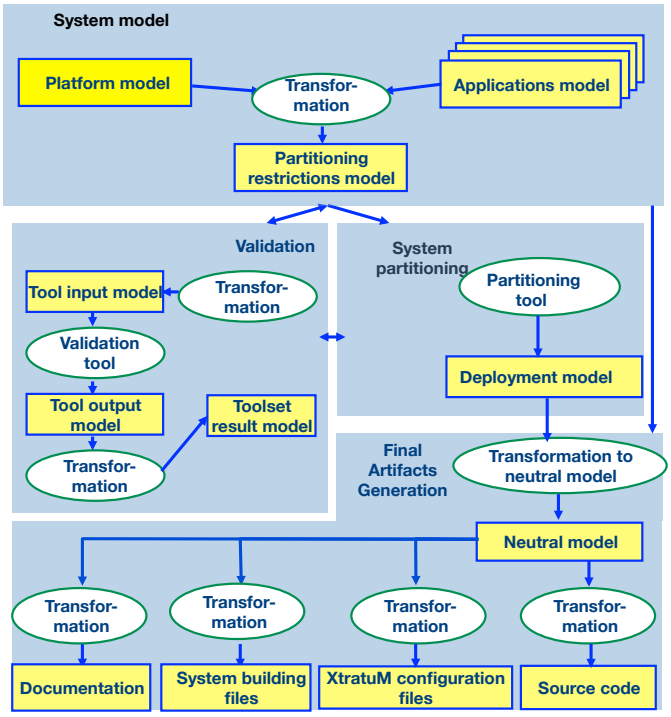

Figure 1: Overall architecture

System modelling: It comprises the main input to the tool. It is composed by three models for describing the execution platforms, the applications, and the restrictions to be applied in the partitioning.

Partitioning tool: It is in charge of generating a system partitioning, that is described in the *deployment model*. It includes system partitions, the assignment of applications to partitions, and the characteristics of the partitions, including the operating system, processor time, memory, etc. The partitioning tool takes as input the system model. It has to consider information, such as the applications' criticality level, their required operating system and hardware devices, etc. Based on this information it generates a deployment model that meets the restrictions and some basic requirements.

Validation: Full correctness of a system partitioning may require complex checks that are difficult to integrate within a single tool. In addition, it is desirable for the toolset to be extended for supporting additional non-functional requirements. It is convenient to be able to use external validation tools that check the correctness of the system configuration with respect to a given criteria.

Generation of final artefacts: when the system partitioning is correct a number of transformation tools generates a set of outcomes that are necessary for creating and building the final system:

- XtratuM configuration files
- System building files.
- Source code skeletons.

This toolset is currently under development. There is a working version that is able to handle simple models. Complexity is being added gradually. The toolset is being developed based on the Eclipse Modelling Tools. Model to model transformers are programmed in Query View Transformation Language (QVT). Model to text generators are based on Acceleo MTL. Metamodels are created using eCore.

4 Toolset extensibility

The MultiPARTES toolset has been designed for being easily extended and evolved. This has been a driver in the design of the architecture, shown in the previous section. There are a number of ways of enriching the current functionalities provided by the toolset, as adding support for additional non-functional requirements, validation tools, or tools for supporting system deployment.

The aim of this section is to describe the basic means for performing toolset extensions, as those mentioned. In fact, these facilities have been the basis for integrating in the core toolset the contributions developed by the partners in the project.

Toolset extension can be done at four main levels:

- System model level: to include in the model annotations for different non-functional properties, or other system aspects.

- Partitioning level: to convert annotations that have to do with partitioning into partitioning constraints.
- Validation level: to use external tools, a deployment model can be validated, according to the system model semantics, as stated by non-functional requirements annotations.
- Generation level: to generate code compliant with non-functional properties annotation or specific configuration parameters for XtratuM.

4.1 Toolset extension at model level

System models can be annotated with information related with non-functional requirements. This is the case with the application model. Initially, all application models include information for partitioning and artefacts generation, such as criticality level or resources needed. Modelled applications rely on the class model in UML2 [8] for its description. The initial version of the toolset relies on the UML-MARTE [9] profile for describing time and resource requirements. In this case, it is possible to model real-time entities (tasks, protected objects) and real-time requirements and parameters. Application resource needs are derived from those of the individual entities.

Following this basis, additional annotations with respect to useful information for the developer can be added. If the information is associated to the application as a whole, then it can be enriched with annotations describing these new aspects. For example, it could be possible to mark an application as being of a specific type that requires specific handling by other tools.

In other cases, the annotations have to be made at the level of application components, such as classes, packages or threads. This case is more demanding. It requires the definition of a profile or metamodel, for defining the way and properties to be specified. Once again, other tools will have access to this information for performing their functions.

4.2 Toolset extension at partitioning level

The partitioning tool is in charge of generating a system partitioning that is consistent with the policies for the different non-functional requirements. The proposed approach is to use the partitioning restrictions model as the basis for the integration of policies of different nature. For each non-functional property or developing aspect, a restrictions generator can be provided. It takes as inputs the platform and applications models, and generates a set of restrictions that ensures that the final system partitioning will meet the constraints in the policies. It is important to point out that the implementation language of the generator is not defined. Anyone can be used, provided that it generates valid restrictions, according to the provided meta-model.

Once all the restrictions derived from the different non-functional properties generators are available, the partitioning tool produces a system partitioning (deployment model) that is compliant with them, if one there exists.

4.3 Toolset extension at validation level

The toolset allows the integration of additional validation tools. This can be required for supporting a new non-functional requirement, or performing a specific validation required in the development of a given system. The inputs to a new validation tool are system and deployment models. The outputs of the validation tool indicates to the partitioning tool whether the proposed deployment model is valid, and a set of new restrictions for driving its correct generation. It may be necessary to include new transformers for generating the validation tool input model or converting the corresponding output, for its integration in the toolset.

4.4 Toolset extension at generation level

The aim is to allow the generation of additional artefacts useful for the development team. As mentioned above, currently the toolset generates XtratuM configuration files, system building files, and source code (Ada skeletons). However, there are additional artefacts that may be generated automatically, such as documentation or files for testing purposes.

Currently, the transformation components in the core toolset take as input the neutral model, for simplification purposes. However, new tools can access other models in the system, such as the system model or deployment model. A new transformation component can access this information for generating the desired artefacts. It is needed to ensure that the required information for this job is included in the models. This integration has to be performed at system model level. The toolset provides means for invoking them and their behaviour will be independent of other transformers, ensuring a straightforward integration.

5 System partitioning

The purpose of this section is to describe the general algorithm taken in the toolset for generating a feasible and automatic system partitioning. This component takes as input the system model: platform model, applications models, and partitioning restrictions model. This one is of particular interest, as it compiles restrictions that must be fulfilled by the resulting partitioning. They can be grouped in two types:

- Explicit: The developers and system integrator define this type of restrictions, which response to specific requirements. As instance, they can define specific hardware devices that must be used by an application, force specific allocations of application into partitions defined by the system integrator, etc.
- Implicit: They are automatically deduced from the system model. As mentioned in section 4, these restrictions are intended to ensure the fulfillment of non-functional requirements specified in the model. As instance, two applications with different criticality level cannot be in the same partition.

The output of this tool is a deployment model, which defines the system partitioning. It includes the description of the partitions. Each of them is characterized by the allocated applications, the used operating system, and required hardware resources.

The global approach for system partitioning relies on the divide and conquer principle. The complexity of this problem, and the requirements for extensibility are additional reasons for this approach. In consequence, the problem is broken down into four stages:

- Allocation of applications into partitions: The aim is to allocate all applications to partitions, trying to minimize its number, while fulfilling the restrictions.
- Allocation of partitions on processor cores. The result of this stage must meet the restrictions related with hardware devices.
- Cyclic plan scheduling design: As mentioned above XtratuM temporal isolation relies on a cyclic scheduling policy that is statically defined. The aim of this stage is to generate the cyclic plan, taking into account application allocation to cores, and applications CPU needs. These are defined in the applications model.
- Validation of the deployment model: Finally, it is possible to validate the resulting system partitioning with respect to general or non-functional requirements. This activity is performed by external tools that can be easily integrated in the toolset. For instance, a response time analysis tool is to be used [4]. Its aim is to ensure that time requirements are met by the proposed partitioning and scheduling plan.

The two initial stages are instances of the general allocation problem. It is NP-Hard, which means that there is no known algorithm that resolves it in polynomial time. This problem has been soundly researched for a long time. After making an analysis of some available options, the allocation problems on the partitioning algorithm in this toolset are based on the greedy algorithm of Iterated Register Coalescing (IRC) [3].

The IRC algorithm is based on colored graph theory. The allocation problem is modeled in a graph, where nodes represents the entities to allocate, colors are the allocation resources, and vertices represent restrictions. Originally, it was intended to help in the allocation of variables into hardware registers for code generation. There are a number of similarities, such as the existence of a number of restrictions that must be followed. This algorithm has been selected due to its good balance between the quality of the results and the implementation complexity.

The use of this approach for allocating applications on partitions required some adaptations. In the proposed solution, nodes represent applications, colors stand for partitions, and vertices are restrictions. The original IRC algorithm assumes a fixed number of resources (registers). However, in this allocation case, the number of resources (partitions) is not limited. Then, the proposed algorithm generates new colors when the allocation is not feasible or additional solutions are required.

The developed algorithm for the allocation of partitions to cores has also been adapted. The aim has been to prioritize solutions where the cores workload is balanced.

In addition, both algorithms have been improved with respect to the original IRC, in order to generate alternative solutions. A proposed system partitioning at this point may be invalid. This can be caused by not being able of generating a feasible cyclic plan or by failing in the validation stage. Then alternatives partitioning are generated, if it is feasible.

There is a working version of the two initial stages, which has been successfully tested with a number of system models. Work is ongoing for performing a more exhaustive and systematic test of these algorithms. There is a very simple version of the cyclic scheduling plan generator, that has been used in simple systems. A more advanced algorithm is currently under development.

6 Conclusions

This paper describes a toolset for supporting the development of mixed-criticality multi-core embedded systems. It relies on the XtratuM hypervisor that provides spatial and temporal isolation, as well as a number of additional features suitable for the development of this type of systems. The presented toolset has been designed according to a set of requirements produced by experts from academia and industry, with knowledge on a number of application domains.

Currently, the toolset provides most of the mentioned functionality, but for simple systems. Support for more complex systems is gradually being included. Future work includes the integration of improved support for time, safety and security, and improvements on the partitioning algorithm. The toolset is being validated in three different use cases: a wind-power turbine control system, the onboard software of the UPMSAT2 satellite, and a video surveillance system.

Acknowledgment

The work in this paper is partially funded by FP7 STREP MultiPARTES project, no 287702 (www.multipartes.eu). The wish to thank the MultiPARTES consortium for its collaboration and help.. The work in this paper has also been funded by the Spanish Ministerio de Educación, Cultura y Deporte, project HI-PARTES (High Integrity Partitioned embedded systems), TIN2011- 28567-C03-01 in the Plan Nacional de I+D+i.

References

[1] Alonso, A., Salazar, E., de Miguel, M.A., A Toolset for the Development of Mixed-Criticality Partitioned Systems, in 2nd Workshop on High-performance and Real-time Embedded Systems, 2014, Vienna, Austria

[2] ARINC: Avionics Application Software Standard Interface ARINC Specification 653-1 (October 2003)

[3] George, L., Appel, A.W.: Iterated register coalescing. TOPLAS 18(3), 300–324 (1996).

[4] González Harbour, M., Gutiérrez, J.J., Palencia, J.C., Drake, J.M.: "MAST modeling and analysis suite for real time applications". In: Proceedings of 13th Euromicro Conference on Real-Time Systems,(June 2001).

[5] M. Masmano, I. Ripoll, A. Crespo, S. Peiro. XtratuM for LEON3: an OpenSource Hypervisor for High-Integrity Systems. Embedded Real Time Software and Systems (ERTS2 2010), May 2010.

[6] MultiPARTES: Multi-cores Partitioning for Trusted Embedded Systems, Available: www.multipartes.eu

[7] MultiPARTES project, "Requirements Platform and Methodology Viewpoint", Deliverable D2.2, http://www.multipartes.eu.

[8] OMG Unified Modeling Language (UML) (2011), http://www.omg.org/spec/UML/2.4.1/, version 2.4.1

[9] OMG UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems (2011), http://www.omg.org/spec/MARTE/, version 1.1

[10] S. Peiro, M. Masmano, I. Ripoll, and A. Crespo. "PaRTiKle OS, a replacement of the core of RTLinux", in Proc. of the Real-Time Linux Workshop, 2007.

[11] E. Salazar, A. Alonso, M.A. de Miguel, J.A. de la Puente. "A Model-Based Framework for Developing Real-Time Safety Ada Systems". In H.B. Keller, et al (eds.), Reliable Software Technologies — Ada-Europe, LNCS 7896, pp. 126–141. Springer-Verlag, 2013.

[12] Schmidt, Douglas C. "Guest editor's introduction: Model-driven engineering." Computer 39.2 (2006): 0025-31.