A Toolset for the Development of Mixed-Criticality Partitioned Systems

Alejandro Alonso, Emilio Salazar, and Miguel A. de Miguel

Dept. de Ingenería de Sistemas Telemáticos
Universidad Politécnica de Madrid, Spain
Email: {aalonso, esalazar, mmiguel}@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, time requirements granularity, etc. MultiPARTES [11] toolset relies on Model Driven Engineering (MDE), which is a suitable approach in this setting, as it helps to bridge the gap between design issues and partitioning concerns. MDE is changing the way systems are developed nowadays, reducing development time. In general, modelling approaches have shown their benefits when applied to embedded systems. These benefits have been achieved by fostering reuse with an intensive use of abstractions, or automating the generation of boiler-plate code.

1 Introduction

The increasing power of processing hardware makes it possible to integrate system functionality in just use one processor, instead of using several ones. Although this has a number of advantages, it presents a problem when developing complex embedded systems. It is common that these systems include applications with different criticality level that previously were executed on different processors, preventing undesirable interferences between them. However, this property is not guaranteed when they coexist in the same processor. 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 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 mixed-criticality embedded systems on multicore platforms. The approach is based on an open-source virtualization layer, provided by the XtratuM hypervisor. A software development methodology and its associated tools will be provided for creating trusted real-time embedded systems to be built as partitioned applications, in a timely and cost-effective way.

In this paper, the MultiPARTES toolset is 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 System model

The aim of the work presented in this paper is to support the development of mixed-criticality embedded systems running on heterogeneous multi-core processors. The coexistence of applications with different criticality level relies on hypervisors that provide virtual machines or partitions, where applications can run with space and time isolation. In this way, it is possible to ensure that applications on different partitions can run independently.

In the context of this work, the system is composed by a set of applications that run on an execution platform. An application is considered to be a software entity that provides a closed set of functionalities. It can interact with other applications for performing their duties. An application is composed by its requirements specification, design models, computer program, and documentation. The execution platform is composed by the hardware devices, a hypervisor, and the set of operating systems that can be run on top of the hypervisor. The hardware platform comprises all computational devices needed by the final system.

The hypervisor provides a set of partitions or virtual machines, where applications are run on isolation. A partition is characterized by the assigned resources, operating system, and a set of applications. The resources assigned should be sufficient for running the applications with the required performance and time requirements. The available CPU is usually characterized by a certain amount of usage time that is replenished periodically. The partition receives a certain amount of memory that is not shared with others. It can also have hardware devices that are not sharable, and that are required by its applications.

XtratuM is the selected hypervisor [18] [10]. It 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. However, this limits the operating systems range that can be used in a particular system.

XtratuM has been designed for meeting a set of properties oriented towards system certification, such as predictability, security, confidentiality or fault isolation. In the context of this work, spatial and space isolation are very relevant. A partition cannot access the memory of another and partitions are executed at specified and fixed temporal intervals. Partitions scheduling is based on a cyclic scheduling policy, that it is statically generated. It precisely states when each partition has to be executed. XtratuM also supports multi-core processors.

3 Toolset Requirements

The development of the toolset has been driven by a requirements specification that has been compiled from different sources. An important input has been the MultiPARTES requirements specification [12] that drives the project evolution. They were mainly defined by the consortium, that 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 other experts in the projects Advisory Board that cooperate with members of the consortium in research activities. 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. This property has to be considered in all the operations performed by the toolset. This is the case when generating a system partitioning. An application with a high criticality level cannot execute in the same partition than another with a lower level. Validation activities and code generation are also influenced by this property. When dealing with an application with a high criticality level, it is common that the source code must meet a number of guidelines for ensuring, for example, high testing coverage or preventing use of dynamic memory.

Support for non-functional requirements. Non-functional requirements are of great importance when dealing with embedded systems. They are not directly associated with an specific function or component of the system. They usually apply to the system as a whole. Non-functional requirements are usually defined as constraints on the system functionality. In the context of this paper, time, safety, and security, are examples of non-functional requirements that will be present in most of the targeted systems. The toolset has to provide means for specifying them, and validating their fulfilment. The toolset can include tools for validating a certain outcome with respect to a non-functional property. The generators and transformers have to consider them, in order to ensure that their outcomes are compliant.

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, non-functional requirements, and hardware resources availability.

Support for multi-core architectures. The execution platform can be multi-core, as it is commonplace in current industrial systems. It should be supported modelling multi-core systems and assigning partitions to cores, according to the model defined by the hypervisor

System modelling. The toolset has to provide means for modelling the whole system, which includes the applications, platform, and any other information that the user has to provide. This is required for ensuring a consistent and coherent handling of all the information related with a system development. It is also required to support legacy applications. This is mandatory for integrating applications that have been developed with other approaches.

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 tools shall support system deployment by generating mechanisms for the automatic building of the system. System deployment also requires the configuration of XtratuM, which includes information on the systems partitions, resources associated to them, etc. This system description can be naturally generated by the toolset, if the required data is provided by the developer.

4 Description of the toolset

4.1 General Approach

Model Driven Engineering (MDE) [21] is a software development approach managed by the Object Management Group that allows engineers to raise the abstraction level of the languages and tools used in the development process. It also helps designers to isolate the information and processing logic from implementation and platform aspects. A basic objective of MDE is to put the model concept on the critical path of software development. This notion changes the previous situation, turning the role of models from contemplative to productive.

Models provide support for different types of problems:

- Description of concepts.
- Validation of these concepts based on checking and analysis techniques.
- Transformation of models and generation of code, configurations, and documentation.

Separation of concerns avoids confusion raised by the combination of different types of concepts. Model-driven approaches introduce solutions for the specialization of the models for specific concerns, as well as the interconnection of concerns based on models transformations. It improves communication between stakeholders using the models to support the exchange of information. The separation of concerns often requires specialized modelling languages for the description of specific concerns.

Another goal of MDE is developing portable, interoperable, maintainable, and well-documented software. MDE is backed up in the separation of system's function specification and how the system uses the resources provided in the underlying platform.

When the UML2 standard was approved [15], the OMG realized that it was necessary to define a profile for real-time and embedded systems. The new profile was called Modelling and Analysis of Real-Time and Embedded [16]. MARTE is a profile for providing support for modelling and analysing real-time and embedded systems. It includes several packages for describing non-functional properties, as well as some secondary profiles for different kinds of systems. It was designed to be compatible with already existing profiles for quality of service and fault tolerance that provide support for annotating embedded systems issues, such as energy consumption, memory, etc. Its main goals are:

- Improve the communication between developers by providing a common modeling environment and methodology for both hardware and software
- Improve the degree of interoperability between tools pertaining to different domains and devoted to different development stages, like specification, design, verification, code generation, etc.
- Leverage the use of models to obtain better analysis and predictions, taking into account both hardware and software characteristics.

4.2 Toolset Architecture

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

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 represented by the *deployment model*, which defines 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

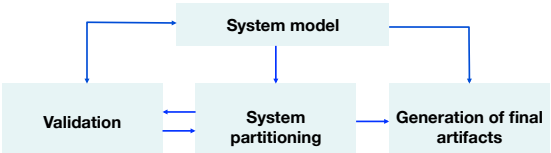

Fig. 1. Overall architecture

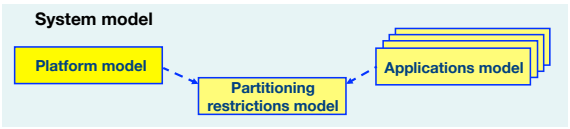

Fig. 2. System modeling

requirements. It is convenient 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 and source code skeletons.

These components are described below with a higher detail level.

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.

4.3 System Modeling

This component includes all the models needed for describing a given system. In general, three types of models are used for describing the execution platform, the set of applications in the system, and restrictions imposed to the partitioning. Figure 2 shows graphically this structure. In the figures of this paper, squares represent data, while ellipses refer to tools. Solid lines refer to data flows and dotted lines mean references to the model.

The execution platform is composed by the hardware, hypervisor, and available operating systems. The description of the hardware has to include all the information required by XtratuM. It will be used for the generation of the deployment outcomes. The execution platform is modelled with an specially designed meta-model [13]. It allows the description of the mentioned entities and ensures that the required information for the hypervisor configuration is provided.

The model for applications can take two forms:

- *Fully modelled*: The full model of the application is provided. This is compliant with a pure model-driven engineering approach. The description of the application is performed using UML2 modelling notation. This is a pure structural and funcional description.
- *Partially modelled:* The application is represented by the source code or the final executable. It is suitable for legacy applications. The application is described with global information. In particular, it is only mandatory to include information on the application criticality and hardware resources needed.

These models can be enriched with information related with non-functional requirements, which are included as annotations in the model. This approach can be easily extended. If it is required to support an additional non-functional property, the modelling language can be extended with the required annotations.

Currently, it is possible to include global information on the applications, and annotations related with real-time requirements. The global information includes information on the criticality of the application, and global resource needs, if it is partially modelled. It is specified as the required memory and CPU percentage, which is defined as a computation time and a replenishment period. This information allows system partitioning to allocate them to the partition when the application is run, to ensure a smooth execution.

The toolset allows for defining real-time annotations for fully modelled applications, based on the UML-Marte profile. Using this profile, different entities can be declared, such as periodic, sporadic or aperiodic tasks, or synchronization objects. In addition, meaningful parameters can be defined, such as periods, deadlines, scheduling policies, priorities, and worst execution times. The toolset extracts automatically the global resource needs for these applications.

Platform and applications models are independent of a particular system. It should be possible to reuse them for creating different systems. The restrictions model includes information that relates them for a particular system or specific criteria for partitioning. In particular, the types of restrictions that are currently supported includes: application that must be allocated on a given partition, application that must (not) be in the same partition than another one, application that requires a particular hardware device, or partition or application that must run on a given core.

Finally, these restrictions can be provided by the user or automatically generated by specific tools that are able to analyse a type of model annotations. In this case, restrictions will be used for ensuring that a system partitioning is compliant with the annotations related for the non-functional property. As an example, this is the case with criticality level annotations. A number of restrictions can be automatically derived for ensuring that applications with different criticality level are not allocated in the same partition.

4.4 System partitioning

This component generates a system partitioning, where each partition is characterized by the applications it includes, the operating system, the processor where it is executed, and the assigned resources. They include a CPU share, memory, and other devices required by its applications. System partitioning is described in the deployment model. This is an internal metamodel, that represents this information and that is not supposed to be edited by the developer. The structure of this component is depicted in figure 3.

A successful partitioning has to met a number of requirements, being the following the most relevant: (i) all applications must be allocated to partitions (ii) partitioning restrictions has to be met, and (iii) resources assigned to partitions must not exceed those available.

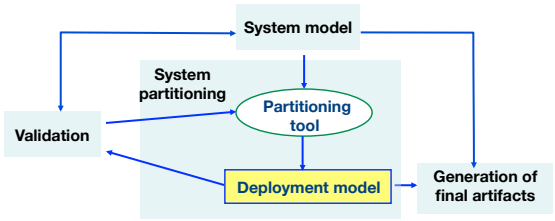

Fig. 3. System partitioning

XtratuM schedules partitions following a cyclic executive policy. The partitioning algorithm has to generate a plan that is repeated cyclically. It defines execution slots and assigns them to the partitions. The size of the slots assigned to a partition has to be compliant with the required processing capability described in the application model.

4.5 Validation

The partitioning algorithm cannot ensure the fulfilment of any type of non-functional requirement, as specific analysis is required. For this reason, the toolset provides a method for including validation tools that checks a given partitioning with respect to a type of non-functional requirement. The toolset provides an approach for automating this process, that is sketched in figure 4.

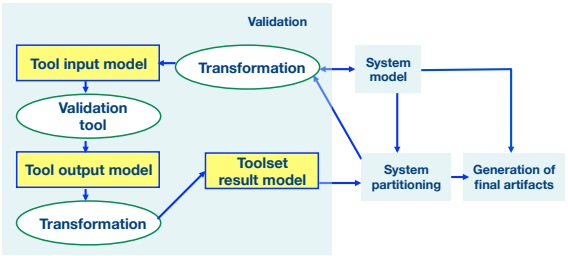

Fig. 4. Validation

The automatic execution of the validation for a system property relies on metamodels and transformers. Initially, a transformer generates a model using the input format of the tool. It is obtained by extracting the relevant information from the system and the deployment models. Then, the validation tool is executed and generates an outcome with the analysis results. Another transformer converts this information on a model that can be used by the partitioning tool, or on a set of new restrictions.

This approach can be used for analysing the fulfilment of time requirements on fully modelled applications. The first transformer can produce the input to

the response time analysis tool, by extracting real-time annotations from the system model and the cyclic plan for partitions, from the deployment model. The tool will perform the analysis and generates an output that can be fed back to the tool. This approach has been successfully used in the toolset developed in the CHESS project [3], which is the predecessor of the one described in this paper. MAST [6] is the selected tool for the response time analysis.

4.6 Generation of final artefacts

This component operates when a validated system partitioning has been built. Its main goal is to generate a number of artefacts that facilitates the building of the final executable system or required documentation. Figure 5 depicts the structure of this component. Initially, it is generated a neutral model, which is a simplified version of the system including only the strictly necessary information for the final outcomes. It simplifies the generation transformers. In particular, it has served to reduce the development cost of transformers for two different programming languages: Ada and Real-Time Java. Currently, the toolset generates three types of artefacts:

XtratuM configuration files: The configuration of the XtratuM hypervisor is stored in a set of files. They describe the number of partitions, the resources assigned to each of them, memory addresses where different software parts must be stored, etc. Their manual construction requires providing a number of tiny details that must be carefully reviewed after each minor change in the system. The MultiPARTES toolset simplifies this process. System models include annotations for providing all required information for creating those files. The toolset checks that no data is missed. Finally, a transformer automatically generates these configuration files.

System building files: The generation of the final system requires locating information on the applications' code, the used operating systems, hypervisor executable, configuration files, etc. In addition, a number of operations are required, such as compiling source code and linking the whole system. The toolset supports the generation of scripts for automating this process. The system model includes slots for providing information such as the location of the sources, the directories where information is located and has to be stored, etc. A script is

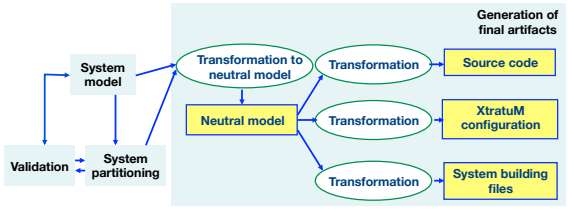

Fig. 5. Generation of final artifacts

provided for building the image that will be loaded and executed in the hardware platform.

Source code skeletons: Application models can be taken as the basis for generating source code. Depending on the detail level of the model, the generated code can be more or less complete. The Eclipse toolset provides few support for the manipulation and generation of Ada source code. Hence, the toolset is able to generate skeletons that correspond to the model entities. System models and the code generator are oriented towards the generation of high-integrity Ada. The Ravenscar computational model [2] is the basis for the application structure. The code generated is compliant with this profile and with the a language subset suitable for this type of systems [8].

Ada code generation takes as input the application class diagram using UML 2.2, including real-time annotations with the UML-MARTE profile. The packages structure, their use relations, tasks, synchronization objects, and passive packages are generated. As an example, if the system model includes a periodic task, the skeleton includes the defined attributes, the activation scheme based on the period and phase in the model, exception handlers, hooks for the functional code, etc. The details of the code generator are described in [20].

This scheme can be easily extended. Currently, it is being designed a generator of evidences needed for the certification of safety critical code. These evidences can include outcomes, such as the schedulability analysis, means for fault detection and contention, results of the validation tools.

5 Related work

The presented framework has three main aspects: automatic generation of a suitable partition schema, integrated analysis and artifacts generation. Automatic code generation tools are widely used nowadays. There are industrial approaches such as IBM Rhapsody [5], which is able to generate complex Ada code from state machines. However, it does not support MARTE models nor the generation of Ada Ravenscar code.

Another alternative is Papyrus [9]. Ada 2005 code compliant with the Ravenscar profile is supported, as well as UML-MARTE models and schedulability analysis. However, Papyrus lacks support for automatic partition and partition artefacts generation. Ocarina [7] [14] is a tool that takes as inputs AADL models and generates Ada and C code. It is able to perform a number of analysis to the input models such as hardware checks, schedulabilty analysis using Cheddar or WCET analysis with Bound-T.

The toolset generated in the context of the ASSERT project has a number of commonalities with that described in this paper. Two sets of tools were developed in this project, one based on HRT-UML [17] [1], and the other one on AADL which later evolved to the current version [19]. Both can generate Ravenscar Ada code and include timing analysis with MAST. TASTE has also been extended for generating Xtratum configuration files [4].

The main differences between these toolsets and the presented toolset are:

– It generates a valid partitioning, which is compliant with the input models.
– It uses industrial standards, instead of ad-hoc adaptions of UML. It relies on UML2, MARTE, and open and free tools and languages (Eclipse, QVT, MTL and Ada).
– It integrates the schedulability analysis, code generation, partitioning generation and XtratuM configuration artefacts generation.

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 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.

The toolset supports a number of fundamental activities, including system modelling, system partitioning, validation, and final artefacts generation. System description allows for the specification of non-functional requirements that are considered along the development process. An innovative aspect of the tool is the support for system partitioning that is compliant with system model and that can be validated with suitable external tools. The generation of code skeletons, hypervisor configuration files and system building files simplifies the construction of the final executable code.

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 safety and security and the design of a more complex partitioning algorithm.

Acknowledgment

The work in this paper is partially funded by FP7 STREP MultiPARTES project, no 287702 (www.multipartes.eu). The MultiPARTES consortium comprises: Ikerlan-IK4 (E), Universitat Politécnica de Valencia (E), Technische Universitt Wien (A), Universidad Politécnica de Madrid (E), TRIALOG (F), FENTISS (E), TELETEL (Gr), Visual Tools (E), ALSTOM Wind (E). 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. Bordin, M., and Vardanega, T., "Correctness by construction for high-integrity real-time systems: A metamodel-driven approach." Reliable Software Technologies–Ada Europe. Springer Berlin Heidelberg, 2007.

2. Burns, A., Dobbing, B., Vardanega, T.: Guide for the use of the Ada Ravenscar profile in high integrity systems. Ada Letters XXIV, 1–74 (June 2004)
3. de Miguel, M.A., Salazar, E., Model-based development for RTSJ platforms. In: Proceedings of the 10th Int. Workshop on Java Technologies for Real- time and Embedded Systems. pp. 175–184. JTRES '12, ACM, New York, NY, USA (2012)
4. Delange, Julien, Christophe Honvault, and James Windsor. "Model-Based Engineering Approach for System Architecture Exploration."
5. G. Eran, D. Harel, and E. Palachi. "Rhapsody: A complete life-cycle model-based development system." Integrated Formal Methods. Springer Berlin Heidelberg, 2002.
6. 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).
7. Hugues, Jerome, et al. "From the prototype to the final embedded system using the Ocarina AADL tool suite." ACM Transactions on Embedded Computing Systems (TECS) 7.4 (2008): 42.
8. ISO/IEC DTR 15942, "ANSI Programming Languages - Guide for the Use of the Ada Programming Language in High Integrity Systems"
9. Lanusse, Agnes, et al. "Papyrus UML: an open source toolset for MDA." Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009). 2009.
10. 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.
11. MultiPARTES: Multi-cores Partitioning for Trusted Embedded Systems, Available: www.multipartes.eu
12. MultiPARTES project, "Requirements Platform and Methodology Viewpoint", Deliverable D2.2, http://www.multipartes.eu.
13. MultiPARTES project, "Specification and Models of Platform", Deliverable D5.1, http://www.multipartes.eu.
14. Ocarina: http://aadl.enst.fr/ocarina/#ocarina
15. OMG Unified Modeling Language (2011), http://www.omg.org/spec/UML/2.4.1/
16. OMG UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems (2011), http://www.omg.org/spec/MARTE/, version 1.1
17. Panunzio, M. and Vardanega, T., "A metamodel-driven process featuring advanced model-based timing analysis." Reliable Software Technologies–Ada Europe 2007. Springer Berlin Heidelberg, 2007. 128-141.
18. 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.
19. Perrotin, M., et al. "TASTE: a real-time software engineering tool-chain overview, status, and future." SDL 2011: Integrating System and Software Modeling. Springer Berlin Heidelberg, 2012. 26-37.
20. 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, Springer-Verlag, 2013.
21. Schmidt, Douglas C. "Guest editor's introduction: Model-driven engineering." Computer 39.2 (2006): 0025-31.