Journal of Computer Science 2012, 8 (11), 1932-1939
ISSN 1549-3636

doi:10.3844/jcssp.2012.1932.1939 Published Online 8 (11) 2012 (http://www.thescipub.com/jcs.toc)

A Method for the Development of Platform Models in the Model Driven Architecture Context

[1,2]Inali Wisniewski Soares, [1, 2]Luciane Telinski Wiedermann Agner, [1]Paulo Cezar Stadzisz and [1]Jean Marcelo Simao

[1]Graduate School of Electrical Engineering and Computer Science,
Federal University of Technology Parana (UTFPR),
Av. 7 de Setembro, 3165, 80.230-901, Curitiba, Parana, Brazil
[2]Department of Computer Science
Mid-West State University (UNICENTRO),
Rua Padre Salvador, 875, 85.015-430, Guarapuava, Parana, Brazil

Received 2012-08-23, Revised 2012-09-20; Accepted 2012-11-07

ABSTRACT

The application of the Model Driven Architecture (MDA) approach to the design of embedded software based on Real-Time Operating Systems (RTOS) is encouraged by the fact that such software has a wide variety of platforms. In this way, the creation of methods for the development of platform models that meet such diversity of platforms is necessary. This study proposes a method called PM-MDA for the development of platform models that design embedded software based on RTOS in the context of MDA. In addition, this study defines the swxRTOS metamodel, a UML 2.0 profile for RTOS based on embedded software design. Such profile defines a set of stereotypes to describe Platform Models (PMs) and is intended to generically describe the services provided by a given embedded system platform of the RTOS. This profile promotes the creation of Platform Models, which will be used as input parameters in the model transformation. Due to the inherent complexity in embedded software design and the existence of a wide variety of platform models, new methods that support the development of such software become crucial.

Keywords: UML Profile, Embedded Software, Software Engineering

1. INTRODUCTION

Model Driven Architecture (MDA) is a contemporary software development approach that aims to shift from conventional code-centric software design to model-centric software design (Chong *et al*., 2011). In Software Engineering, the main idea behind the MDA approach lies in the fact that software can be initially built from an abstract representation, which is successively refined until implementation is reached (Hamous-Lhadj *et al*., 2009).

The MDA development process involves: specification of the system functionality with a Platform Independent Model (PIM), specification of a Platform Model (PM), selection of a specific platform for the system, Model Transformation (MT) of a PIM into a Platform Specific Model (PSM) based on a specific platform (OMG, 2010; Lecomte *et al*., 2011; Loniewski *et al*., 2011).

The PM provides a set of technical concepts that represent parts and services of a platform. Moreover, such PM must specify constraints on the use of these elements. MT, in its turn, can be defined as the generation of a target model from a source model based on a set of rules that define the link between its elements (OMG, 2010; Meertens *et al*., 2010).

An embedded system can be defined as an electronic and autonomous system designed to a specific task

Corresponding Author: Inali Wisniewski Soares, Graduate School of Electrical Engineering and Computer Science, Federal University of Technology Parana (UTFPR), Av. 7 de Setembro, 3165, 80.230-901, Curitiba, Parana, Brazil Tel: + 55 41 3310 4759

(Kessaci *et al*., 2011). Due to the advancements in the field of electronics, most applications have been developed by microcontrollers based on embedded systems. This type of system has become very widely spread to all kinds of equipment, either for consumer, industrial or military products (Baskaran and Vijula, 2012). Currently, the growing need for new embedded products with additional functionalities is a tendency, thus demanding the increment of more complex software components in the system. Therefore, the complexity of embedded software systems emphasizes the need for high-level development approaches, such as MDA. MDA approach is appealing to the design of embedded systems, once models can be easily evolved as hardware and software requirements evolve (Tucci-Piergiovanni *et al*., 2011).

UML is the standard language for representing software design models. It provides a rich set of notations, diagrams, extension mechanisms and, whatever its advantages and drawbacks are, it is undeniably one of the most adopted modeling languages of this decade (Bendraou *et al*., 2010). Particularly, the UML core still lacks key artifacts for precisely describing PMs of RTOS execution platforms.

Essentially, software development is an activity that demands intensive human knowledge, involving software developers who execute a software development process by making use of expert knowledge (Omar *et al*., 2011). In the domain of embedded systems, MDA employment is thus even more encouraging due to the wide variety of platforms and the development complexity of such systems. In this context, the present paper aims to contribute to the improvement of the process quality and productivity for RTOS-based embedded software design and mainly focuses on the proposal of a UML profile: swxRTOS-defines a set of stereotypes to describe the PM and proposes a method through which these PMs will be created, allowing the definition of PMs separate from the model transformation process and thus resulting in the transformation of models that are more efficient and adaptable to other platforms.

2. MATERIALS AND METHODS

The Platform Model provides a set of technical concepts that represent the parts and services of a platform as well as the different types of elements that can be employed in the use specification of the platform by an application (OMG, 2010). In order to understand and define a Platform Model, it is necessary to understand the concept of platform. At first, platform was simply defined as "computer hardware that executes software". Later on, such definition included software-based platforms such as operational systems. That is to say, the term platform implies into a set of hardware/software tools that enable the execution of software applications. "Enable the execution" means, in this case, providing external mechanisms or services used by one or more software applications. In this way, a software platform provides a set of general-purpose capabilities (services), e.g., communication tools among processes, memory archive and memory management services (Selic, 2005; Dube and Dixit, 2012).

The services of a platform are typically accessed by an Application Programming Interface (API). The concept of API is essential in Software Engineering as a representation of the information hiding principle. The services that a supplier software asset offers to client applications are exposed through an API, a specification that hides implementation details and describes how to properly use services (at least their signature) and the kind of results they provide (Izquierdo *et al*., 2012). A software platform can thus be displayed as an abstraction mechanism as well. The main purpose of a platform, however, is not found on abstraction, but on enabling the access to its functions (or services). In this case, abstraction is just a convenient way to enable the use of the services offered by the platform.

Understanding that a software platform differs from the applications that it supports is crucial. Although the platform services are used to implement an application, such services are not part of the application. Another feature related to platforms refers to its independence of the applications, i.e., its availability and operation are not dependent on the existence or execution of the applications. Regarding the development of embedded software based on RTOS, the wide variety of platforms is a consequence of the large number of suppliers and technologies available.

Under the scope of this study, therefore, platform is defined as a set of hardware/software tools that enable the execution of software applications based on an RTOS. Hardware consists of the respective hardware platforms of embedded system based on specific processors required for the execution of an RTOS. In its turn, software consists of the RTOS and its respective APIs.

2.1. Platform Independence of Embedded Software

Embedded systems are more affected by the adopted platform due to their incorporated hardware features and the restrictions imposed on them. As a consequence, in order to achieve "platform independence" in these systems, the

platform to be used must be defined in an abstract way, considering its interest attributes. That means, a software project must perform the application modeling based on an abstract platform model (Selic, 2012).

The X Real-Time Kernel (Renaux *et al*., 2010), an example of RTOS defined in this study as a target platform, can be employed in different platforms. Those platforms are identified according to the core used: ARM7TDMI, ARM9, Cortex-M4, Cortex-A8, among others. The X Real-Time Kernel is used in Deeply Embedded Systems.

Improving process quality and productivity for developing embedded software has become a big challenge due to the wide variety of existing platforms. This study aims to contribute with this challenge and mainly focuses on two aspects: (a) a proposal of the UML metamodel for Real-Time Embedded Software application and (b) a method in which this metamodel will be applied, resulting in the creation of platform models separated from the model transformation process.

UML allows the creation of new languages for different purposes. For example, in order to adapt UML 2.0 to different applications and platform domains, extension mechanisms are provided by the language. Extension mechanisms in UML 2.0 can be classified into first-class and second-class extensibility (OMG, 2010; Selic, 2012).

First-class extensibility is handled through the Meta Object Facility-MOF (OMG, 2011). This approach allows modifications in the existing metamodels and the creation of new metamodels without constraints. Second-class extensibility does not allow modifications in the existing metamodels. Instead, it enables adapting metamodels for specific purposes by extending existing metaclasses. Adaptations are defined by using stereotypes, tagged values and constraints, which are grouped in a profile.

A UML profile contains constructs that are specific to a particular domain, platform or method. The profiles proposed in this study utilize only the second-class extensibility, which is considered to be suitable for applying UML 2.0 to embedded software design.

2.2. The swxRTOS Profile

Responsible for generating PMs, aims to facilitate the "platform independence" in the development of embedded software. It must describe an abstraction layer for the RTOS X Real-Time Kernel (defined in this study as an example of RTOS) in a generic way, not considering a specific hardware platform, i.e., a specific processor attached to a specific electronic board. For instance, based on the swxRTOS profile, the following PMs can be defined: (1) PM for X Real-Time Kernel in NXP ARM7 processors and (2) PM for X Real-Time Kernel in Atmel ARM7 processors. The swxRTOS profile is composed of sub-profiles such as the swxCoreRTOS, which represents the basic concepts of the high-level constructs needed to support both concurrency and interactions. In its turn, the ddxRTOS is another example of a sub-profile of the swxRTOS, representing the concepts related to the physical microcontroller peripherals used in the RTOS. Due to length limitation, only fragments of the swxCoreRTOS and ddxRTOS sub-profiles will be considered and described in **Table 1 and 2** respectively.

2.3. PM-MDA Method

PM-MDA is a method for developing Platform Models in the context of the MDA approach. This method can be used as a development guide for Platform Models independently of the transformation rules. In this way, the development of transformation models that are more adaptable in the context of the MDA approach is possible. The steps that constitute the PM-MDA method are represented in **Fig. 1** and described as follows:

2.4. Step 1-RTOS Selection

In this step it is possible to define the RTOS for which the Platform model will be designed. In this study the RTOS X Real-Time Kernel will be used. This RTOS refers to the development of embedded systems with rigid constraints regarding time and computational resources (Renaux *et al*., 2010). The RTOS X Real-Time Kernel is used in embedded systems based on 32-bit RISC processors with ARM architecture, more specifically ARM7 and ARM9.

2.5. Step 2-Defining Hardware and Processor Architecture

After defining the RTOS to be used, the hardware and processor architectures will be determined. There are various kinds of hardware and processors available for usage according to the selected RTOS. For the X Real-Time Kernel, the processors available are: ARM7 and ARM9. Carrying out this step is essential for defining the support to the processors' peripheral drivers. The next step is the composition of the software/hardware features of the Platform Model. To do so, the platform metamodel previously described will be used. In that metamodel, the elements needed and the respective associated standards will be available.

2.6. Step 3-Analyzing the API of the Selected RTOS

In this step, the API of the selected RTOS is analyzed in detail. In so doing, the verification of the details related to the software and hardware concepts identified in steps 1 and 2 is intended. After such analysis, the definition of the elements that form the Platform Model is possible. The following APIs are analyzed in this study: RTOS X Real-Time Kernel versions 1.0 and 2.0.

2.7. Step 4-Platform Model Design

The Platform Model for the selected RTOS is designed in this step, in accordance with the software and hardware features identified and analyzed in the previous steps. Then, the next step consists in composing the software and hardware features of the Platform Model by making use of the platform metamodel, in which the required elements and respective associated standards are available. **Figure 2** illustrates an example of the PM-ARM7 design through the employment of the MPM-swxRTOS. The MPM-swxRTOS specifies two metaclasses: (1) swxCoreRTOS: represents the software features of the RTOS; (2) ddxSerialComm: defines the hardware features of the RTOS, representing a serial port driver of ARM processors' architecture. The PM-ARM7 specifies two classes: (1) X: represents the main class of the RTOS X Real-Time Kernel; (2) CSerialARM7: defines the hardware features of the RTOS X Real-Time Kernel.

Table 1. Stereotypes of the swxCoreRTOS sub-profile

Stereotype	swxCore
Description	Concepts regarding the software description in concurrent execution contexts
Tagged	threadName: Name of a task
Values	threadPriority: Execution priority of a task
	timeSuspension: Time of suspension
	sleepThreadFor: Suspension of a task for a definite period
	activateThread: Creation of a task
	threadStackSize: Number of words of which the task stack is composed
Stereotype	swxSemaphore
Description	Concepts regarding the creation and management of a semaphore
Tagged	num: semaphore initial value
Values	semaph: variable whose values are 0, 1 or negative
	checkActualVlr: checks the semaphore current value
	waitSemaphore: wait state of the semaphore
Stereotype	swxTime
Description	Concepts regarding time values.
Tagged	nanosec: Stores time related values in nanoseconds
Values	sec: Stores time related values in seconds
	time: Stores time related values
	getNanoSec: retrieves the current value of the nanosecond counter

Table 2. Stereotype of the ddxCoreRTOS sub-profile

Stereotype	ddxSerialComm
Description	Represents general information regarding device drivers of the serial ports of ARM processors.
Tagged	configDD: Configures the Device Driver
Values	stateDD: Represents the Device Driver state
	getStateDD: Retrieves the Device Driver state
	startDD: Starts a device after its configuration
	stopDD: Stops a device
	numTypeDD: Represents the device driver type code

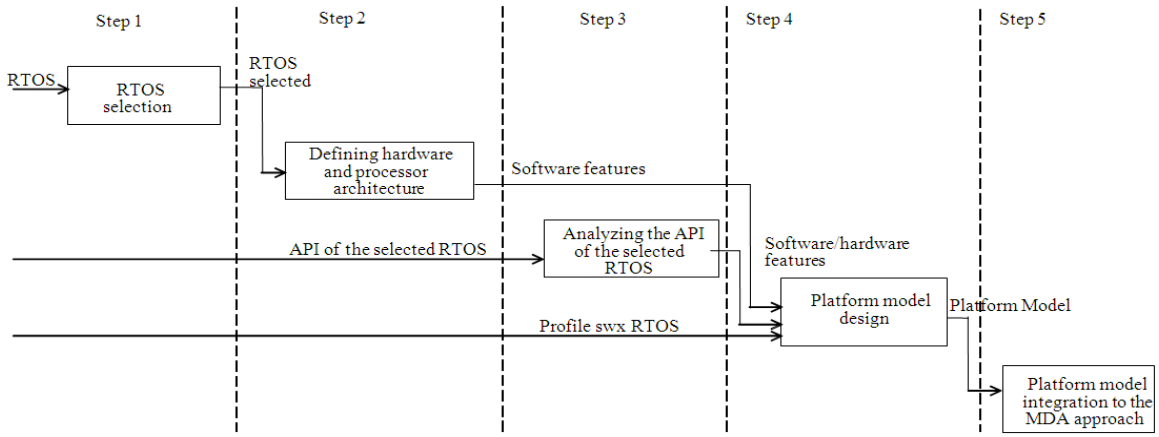

Fig. 1. Overview of the PM-MDA method

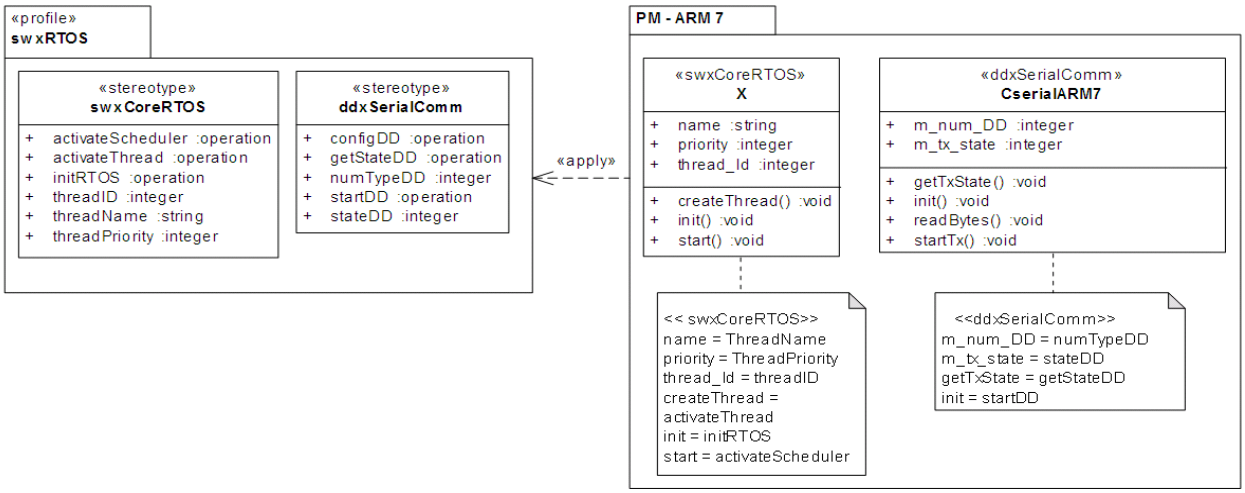

Fig. 2. Example of a PM - ARM7 created through the PM- MDA method

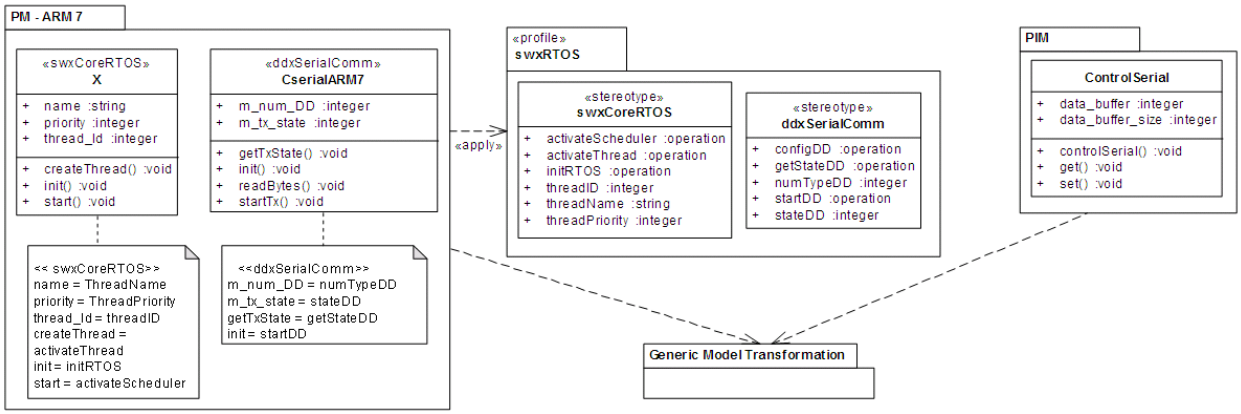

Fig. 3. Example of a PM integrated to the MDA approach

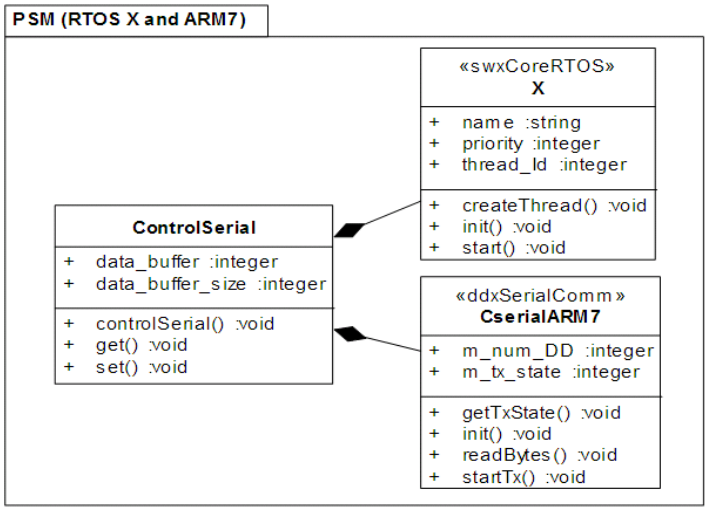

Fig. 4. Example of a PSM created through the abstract PM based on the swxRTOS profile

2.8. Step 5-Platform Model Integration to the MDA Approach

The last step of the PM-MDA method refers to the integration of the Platform Model to the MDA approach. The PIM and PM models are used as input parameters in the model transformation process. In so doing, a model transformation language, e.g., the Atlas Transformation Language (Troya and Vallecillo, 2010), is selected to be used in the creation of a model transformation. A model transformation, as described in (Agner *et al*., 2012), can be used to insert the PM defined in this method and a PIM into a PIM-into-PSM model transformation. Such approach refers to an endogenous refinement of models based on the UML metamodel. In a model refinement most elements of the source model (PIM) are copied to the target model (PSM), while other elements must be changed in order to incorporate platform-specific aspects. Refinement is described as a model transformation process in which more details are added to a previously defined model (Straeten *et al*., 2007).

2.9. Example of an Application of the PM-MDA Method

A simplified example of the PM-MDA Method application is illustrated in **Fig. 3**. In this example, the RTOS selected was the X Real-Time Kernel (step 1), the hardware and processor architecture selected was the ARM7 (step 2), the API analysis of the *RTOS* X Real-Time Kernel version 1.0 was conducted (step 3), the platform model was created by using the profile swxRTOS (step 4). The last step consists in integrating the PM to the MDA approach, as showed in **Fig. 3** as for representing its operation and illustrating the impact of hardware and software aspects in embedded systems development. **Figure 3** illustrates a simple example of generic transformation of models and uses a PIM and a PM model as input parameters, once the PM was defined regarding the profile swxRTOS.

The PIM defines a part of an application model that, in its turn, defines a class for controlling data transmission through the serial port driver, named ControlSerial. PIM is a platform independent model, so the RTOS services are defined in an abstract (generic) way and are not related to a specific platform. The PM was defined regarding the profile swxRTOS for the RTOS in ARM7 processors. **Figure 4** ilustrates an example of a PSM created through a generic transformation approach based on a PM built by using the profile swxRTOS.

3. RESULTS

The present study proposed a method for defining platform models for embedded software based on RTOS. The main objective of such method resides in enabling the development of applications based on RTOS-based embedded software and that are more independent of platform. Platform independence increases productivity in software development, one of the main objectives of Software Engineering.

Using this method allows the creation of platform models that can be inserted into a model transformation, i.e., it is possible to separate platform concerns from

model transformation concerns. The definition of methods for building platform models promotes the improvement of the MDA approach, once model transformations can be generically developed and reused for each platform selected.

In the creation of the platform model, the swxRTOS platform metamodel was used. It consists of a UML profile that is easy to be applied and sufficient to model the artifacts required for the platform models used by the RTOS X Real-Time Kernel, presented as an example of RTOS in the PM-MDA method.

The PM-MDA method meets the highest abstraction level of the MDA approach by adopting a high abstraction level from the metamodel swxRTOS. In addition, the PM-MDA method enables the integration of the platform model to the MDA approach and illustrates a simplified example of such integration.

4. DISCUSSION

Cheng (2010) proposes a model-driven method to describe platform. This method is described with action semantic meta language, including PIM designing, mapping from PIM to PSM and mapping from PSM to application program. However, such work does not take into account the operating system as a platform (Cheng 2010). A model-driven modelling approach allowing describing an execution platform is proposed in (Lafaye *et al*., 2011). Such modelling enables a systematic approach to set platform aspects; however this aprroach does not provide specific artifacts to model RTOS execution resources. This related work does not provide artifacts to produce a code that can be easily interfaced with a platform based on an RTOS.

A metamodel for describing multitask computing platforms is proposed in (Thomas *et al*., 2008). This metamodel works like a language, describing the Application Programming Interface (API) for such platforms. This study proposed the UML Software Resource Modeling (SRM)-sub-profile of MARTE-for RTOS multitask platforms. The model of a platform is considered a model of such interface and thus the platform metamodel as a language for describing that interface. MARTE is a UML profile designed for model driven development of Real-Time and Embedded Systems. However, modeling for a specific RTOS requires adapting the MARTE profile to the modeling conventions of that RTOS and, as a consequence, the modeling will not always be straightforward.

The importance of this research lies in the proposition of a metamodel for designing PMs as well as the proposition of a method called PM-MDA for creating PMs that can be easily applied to a specific platform, using similar nomenclature and semantics of a specific RTOS.

5. CONCLUSION

Embedded software development benefits from the use of the MDA approach due to the inherent complexity of such systems. In RTOS embedded systems, software needs to be integrated with the underlying hardware, so that such integration causes embedded software hard to be developed, analyzed and reused.

In this study, we propose the integration of platform models for RTOS embedded systems into the MDA approach. We illustrated our strategy with the PM-MDA method, which helps the embedded software designer create platform models and, then, integrate them into a generic model transformation approach. For defining the PM-MDA method, the swxRTOS metamodel was proposed. It is used to represent PMs, enabling RTOS services, attributes and the corresponding associated hardware to be used and depicted in a generic way.

In our future work we intend to propose the mapping at the meta-meta level, i.e., from an architectural meta-meta model into MOF, so as to extend the use of the swxRTOS metamodel to specify platform models for other RTOS. Furthermore, we intend to develop an automatic elaboration of the correspondence specification concepts between MDA PIM, MDA PM and MDA PSM metamodels for the model transformation process.

6. REFERENCES

Agner, L.T.W., I.W. Soares, P.C. Stadzisz and J.M. Simao, 2012. Model refinement in the model driven architecture context. J. Comput. Sci., 8: 1205-1211. DOI: 10.3844/jcssp.2012.1205

Baskaran, K. and G. Vijula, 2012. Multicore based open loop motor controller embedded system for permanent magnet direct current motor. Am. J. Applied Sci., 9: 924-933. DOI: 10.3844/ajassp.2012.924.933

Bendraou, R., J.M. Jezequel, M.P. Gervais and X. Blanc, 2010. A Comparison of six UML-based languages for software process modeling. IEEE Trans. Softw. Eng., 36: 662-675. DOI: 10.1109/TSE.2009.85

Cheng, F., 2010. MDA implementation based on patterns and action semantics. Proceedings of the 3rd International Conference on Information and Computing (ICIC), Jun. 4-6, IEEE Xplore Press, Wuxi, Jiang Su, pp: 25-28. DOI: 10.1109/ICIC.2010.100

Chong, S., C.B. Wong, H. Jia, H. Pan and P. Moore *et al*., 2011. Model driven system engineering for vehicle system utilizing model driven architecture approach and hardware-in-the-loop simulation. Proceedings of the International Conference on Mechatronics and Automation (ICMA), Aug. 7-10, IEEE Xplore Press, Beijing, pp: 1451-1456. DOI: 10.1109/ICMA.2011.5985964

Dube, M.R. and S.K. Dixit, 2012. Modeling theories and model transformation scenario for complex system development. Int. J. Comput. Appli., 38: 11-18. DOI: 10.5120/4698-6847

Hamous-Lhadj, A., A. Gherbi and J. Nandigam, 2009. The impact of the model-driven approach to software engineering on software engineering education. Proceedings of the 6th International Conference on Information Technology, Apr. 27-29. Las Vegas, Nevada, pp: 719-724. DOI: 10.1109/ITNG.2009.160

Izquierdo, J.L., J. Frederic, C. Jordi and G.M. Jesus, 2012. API2MoL: Automating the building of bridges between APIs and model-driven engineering. Inform. Software Technol., 54: 257-273. DOI: 10.1016/j.infsof.2011.09.006

Kessaci, Y., M. Mezmaz, N. Melab, E. Talbi and D. Tuyttens, 2011. Parallel evolutionary algorithms for energy aware scheduling. Int. Decision Syst. Large-Scale Distributed Environ., 362: 75-100. DOI: 10.1007/978-3-642-21271-0_4

Lafaye, M., M. Gatti, D. Faura and L. Pautet, 2011. Model driven early exploration of IMA execution platform. Proceedings of the 30th IEEE/AIAA Digital Avionics Systems Conference (DASC), Oct. 16-20, IEEE Xplore Press, Seattle, WA, pp: 7A2-1-7A2-11. DOI: 10.1109/DASC.2011.6096113

Lecomte, S., S. Guillouard, C. Moy, P. Leray and P. Soulard, 2011. A co-design methodology based on model driven architecture for real time embedded systems. Math. Comput. Modell., 53: 471-484. DOI: 10.1016/j.mcm.2010.03.035

Loniewski, G., A. Armesto and E. Insfran, 2011. An Architecture-oriented model-driven requirements engineering approach. Proceedings of the Model-Driven Requirements Engineering Workshop (MoDRE), Aug. 29-29, IEEE Xplore Press, Trento, pp: 31-38. DOI: 10.1109/MoDRE.2011.6045364

Meertens, L.O., M.E. Iacob and L.J.M. Nieuwenhuis, 2010. Goal and model driven design of an architecture for a care service platform. Proceedings of the 2010 ACM Symposium on Applied Computing, (ASAC' 10), ACM, New York, USA., pp: 158-164. DOI: 10.1145/1774088.1774119

Omar, M., S.L. Syed-Abdullah and A. Yasin, 2011. The impact of agile approach on software engineering teams. Am. J. Econ. Bus. Admin., 3: 12-17. DOI: 10.3844/ajebasp.2011.12.17

OMG, 2010. Model driven architecture: The architecture of choice for a changing world OMG.

Renaux, D.P.B., R.E. Goes and R.R. Linhares, 2010. Performance characterization of real-time operating systems for systems-on-silicon. Proceedings of the 12th Brazilian Workshop on Real-Time and Embedded Systems, (BWRTES' 10).

Selic, B., 2005. On software platforms, their modeling with UML 2 and platform-independent design. Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC' 05), Seattle, USA., pp: 15-21. DOI: 10.1109/ISORC.2005.40

Selic, B., 2012. The less well known UML: A short user guide. Proceedings of the 12th international conference on Formal Methods for the Design of Computer, Communication, and Software Systems, (SFM' 12), Springer-Verlag Berlin, Heidelberg, pp: 1-20. DOI: 10.1007/978-3-642-30982-3_1

Straeten, R.V.D., V. Jonckers and T. Mens, 2007. A formal approach to model refactoring and model refinement. Software Syst. Modell., 6: 139-162. DOI: 10.1007/s10270-006-0025-9

Thomas, F., S. Gerard, J. Delatour and F. Terrier, 2008. Software real-time resource modeling. Lecture Notes Elect. Eng., 10: 169-182. DOI: 10.1007/978-1-4020-8297-9_12

Troya, J. and A. Vallecillo, 2010. Towards a rewriting logic semantics for ATL. Proceedings of the 3rd International Conference on Theory and Practice of Model Transformations, (ICMT' 10), Springer-Verlag Berlin, Heidelberg, pp: 230-244. DOI: 10.1007/978-3-642-13688-7_16

Tucci-Piergiovanni, S., C. Mraidha, E. Wozniak, A. Lanusse and S. Gerard, 2011. A UML model-based approach for replication assessment of AUTOSAR safety-critical applications. Proceedings of the IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, (TRUSTCOM), IEEE Computer Society Washington, DC, USA., pp: 1176-1187. DOI: 10.1109/TrustCom.2011.159