Journal of Computer Sciences

Original Research Paper

Application of Platform Models in Model Driven Engineering of Embedded Software

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

[1]*Department of Computer Science,*
Mid-West State University (UNICENTRO), Guarapuava, Paraná, Brazil
[2]*Graduate School of Electrical Engineering and Computer Science,*
Federal University of Technology Paraná (UTFPR), Curitiba, Paraná, Brazil

Article history
Received: 22-06-2015
Revised: 12-12-2015
Accepted: 31-12-2015

Corresponding Author:
Inali Wisniewski Soares
Department of Computer Science, Mid-West State University (UNICENTRO), Guarapuava, Paraná, Brazil
Email: inali@unicentro.br

Abstract: This paper presents two UML 2.0 profiles for designing embedded software based on Real-Time Operating Systems (RTOS). The first profile, named Application Modeling Profile (AMP), defines the necessary elements for the application modeling. The second one, named swxRTOS, defines a set of stereotypes to describe platform models and the mapping between application and platform. These profiles are used for the development of Application and Platform separately from the model transformations in the context of the Model Driven Engineering (MDE) approach.

Keywords: Model Driven Engineering, Profile UML, Platform Model, Embedded Software

Introduction

Model Driven Engineering (MDE) is an approach for software development that focuses on the creation of software models. In the MDE context, each software artifact is considered a model or a model element (Kent, 2002). Thus, MDE emphasizes the role of models as the primary development artifacts by providing a set of guidelines for the definition of models as well as the transformations between them.

The term "platform" refers to any set of hardware or software mechanisms that enable the execution of software applications (Selic, 2005). Throughout this paper, however, the term "platform" denotes Real-Time Operating Systems (RTOS) and their respective hardware platforms of embedded systems, based on specific processors required for their execution.

Particularly, the Platform Model (PM) provides a set of technical concepts that represents services of a platform (Truyen, 2006). Model Transformation is defined as the conversion of a model higher level to a model of the lowest level of abstraction, considering a set of rules well defined (Dube and Dixit, 2012).

In most MDE development approaches, however, the PM is implicitly employed in the model transformations (Lecomte *et al*., 2011). As the features associated with the platform are not separated from the model transformation features, for each selected platform there must be a specific model transformation (Wagelaar and Jonckers, 2005). This means that model transformations are constructed specifically for a given platform and must be reconstructed in the case another platform is envisioned.

Although embedded systems are currently found everywhere, they are considered as complicated and requiring complex artifacts. Actually, for each new embedded product more additional functionality is usually demanded and more complex software components are added to the system. The complexity of embedded software systems thus emphasizes the need for high-level development approaches such as MDE.

Unfortunately, the support provided by MDE to the development of embedded software, mainly RTOS-based software, is still limited (Kukkala *et al*., 2005). MDE primarily focuses on middleware target platforms such as EJB, Web Services, NET and CORBA for ordinary Operating Systems (OS). Although such platforms are applied to ordinary OS, they allow applications to be projected independently nonetheless (Truyen, 2006).

However, a typical embedded software comprises some features like concurrency, real-time processing and limited resources. Such requirements are commonly supported by an RTOS, whereas the use of middleware is more unlikely to meet them (Maeng *et al*., 2006). As a result, many embedded applications are designed and implemented to run directly upon the RTOS.

Furthermore, the OMG has defined the Unified Modeling Language (UML) (UML, 2011) as the standard language for representing software development models (Truyen, 2006). Indeed, the UML core still particularly lacks key artifacts for accurately describing PMs of RTOS execution platforms. The development of embedded software requires the reduction of the gap between hardware and software designs given that the use of MDE is even more advantageous due to the wide variety of platforms in the domain of embedded systems.

This paper aims to contribute to the enhancement of quality and productivity in the RTOS-based embedded software design process, mainly focusing on proposal of two UML profiles: Application Modeling Profile (AMP)-defines the necessary elements for creating the PIM; and swxRTOS-defines a set of stereotypes to describe the PM and the mapping between PIM and PM.

This paper is organized as follows. Section 2 introduces some related work. Section 3 describes the UML profiles. Section 4 presents an example of the exemplification if the AMP profile and the swxRTOS profile. Section 5 concludes this paper.

Related Work

Most model-driven approaches for embedded software have focused on improving specific platform models (Karsai *et al*., 2008; Jeon *et al*., 2009). Platforms are thus not described as input models for the development of model transformations, but indeed platform parameters blend in with transformation rules. As a result, there are few studies related to the development of embedded software using explicit platform models in the MDA context (Sendall and Kozaczynski, 2003; Kolovos *et al*., 2008; Renaux *et al*., 2010).

Kukkala proposed a model-driven methodology to describe applications and platforms. This methodology allows the description of platform structures and also binds applications and platforms. However, this work does not take into account the RTOS as a platform (Kukkala *et al*., 2008).

Selic describes a general UML profile for platform modeling and deployment of relationships between platforms and applications (Selic, 2005). This model enables a systematic approach to set platform aspects. However, this profile does not provide specific artifacts to model RTOS execution resources. In addition, it does not provide artifacts to produce a code that can be easily interfaced with a platform based on an RTOS.

The UML profile Modeling and Analysis of Real-Time and Embedded systems (MARTE) (MARTE, 2008) provides a dedicated sub-profile for the development of Real-Time Embedded Systems, called Software Resource Modeling (SRM), which permits the description of the RTOS services. Nevertheless, RTOS-specific modeling requires the adaptation of the MARTE profile to the modeling conventions of the RTOS considered. In the example presented in this study, the RTOS X Real-Time Kernel (Renaux *et al*., 2010) and the ARM7 processors are employed as a platform. Thus, the modeling will not be easily executed and most importantly, it will not be sufficient to all elements of such platform.

Proposed UML Profiles

UML allows the creation of new languages for different purposes. For example, extension mechanisms are provided in order to adapt UML 2.0 to different applications and platform domains. The adaptations are defined by using stereotypes, tagged values and constraints, which are grouped in a profile (UML, 2011).

Stereotype is a kind of class that extends classes through extensions. Just like a class, a stereotype may have properties, which may be referred to as tag definitions. When a stereotype is applied to a model element, the property values may be referred to as tagged values. A constraint can be attached to any model element to refine its semantics. A constraint can be defined by means of an informal explanation in natural language and/or by means of Object Constraint Language (OCL) (Warmer and Kleppe, 2003) expressions. The OCL is a formal language for the analysis and design of software systems. It is the subset of the UML standard that allows software developers to write constraints and queries over object models.

AMP Profile Overview

The Platform Independent Model (PIM) must achieve a sufficient level of platform-independence, enabling its transformation into Platform Specific Models (PSMs), according to the selected platform. This paper thus proposes the employment of the Application Modeling Profile (AMP) in the creation of the PIM, given that it allows software engineers to design this model regardless of the detailed knowledge on the selected platform.

The AMP profile specifies two stereotypes, which are extensions of the “UML operation” metaclass:

- <<rtDDoperation>> stereotype: Represents the service operations of RTOS device drivers (e.g., display-or keyboard-related services)
- <<rtSWoperation>> stereotype: Represents the service operations of RTOS kernel (e.g., thread-or interrupt-related services)

Consequently, the PIM model will include elements from the AMP profile, which abstractly defines RTOS services. The main purpose of using this profile resides in annotating services in the PIM without any reference to a specific platform. Finally, the PIM (based on a selected platform) is transformed into a PSM in the context of the MDA approach by means of a model transformation process.

swxRTOS Profile Overview

The swxRTOS profile is an adaptation of the UML metamodel responsible for the creation of Platform Models. This profile aims to facilitate "platform independence" in the development of embedded software. Also, it defines a set of stereotypes in order to abstractly describe the services provided by the RTOS-based platform as well as their respective hardware platforms of embedded systems and the specific processors required for their execution.

The X Real-Time Kernel is an example of RTOS, presented in this study as a target software platform. This kernel can be employed in different platforms, identified according to the microprocessor used: eLPC-Main 2122, eLPC48, eLPC64, eLPC 144, eAT55 (eSysTech, 2007).

A sample of some PMs based on the swxRTOS profile is pointed out as follows: (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 (Fig. 1) is composed of the following sub-profiles:

- swxCoreRTOS: Represents the basic concepts the high-level constructs needed to support both concurrency and interactions
- swxTimeRTOS: Identifies the set of time-related concepts and semantics
- ddxRTOS: Represents the concepts related to the physical microcontroller peripherals used in RTOS X Real-Time Kernel

The templates illustrated in Table 1 are used to detail the swxCore RTOS, swxTime RTOS and ddxRTOS sub-profiles, respectively. Such templates are based on the UML 2.0 specification (UML, 2011) and on (Rosado *et al*., 2011). It is important to point out that in those templates only some of the elements of the sub-profiles are described. These templates allow us to describe the abstract syntax of the elements of these profiles in natural language. The model elements are described below:

- Stereotype: Name of the stereotype
- Description: Indicates the purpose and significance of using stereotypes
- Anotation: Corresponds to an icon associated to the stereotype, its graphic notation
- Tagged values: Are used to define different properties of the RTOS X Real-Time Kernel platform models and their respective hardware platforms describes the properties of the stereotype, i.e., their attributes and operations
- Name: Name of the tagged value
- Description: Descreve of the tagged value
- Type: Type of the attribute or operation
- Constraints: Correspond to a set of limitations with regard to the stereotypes and their relation with other stereotypes and with UML elements. Some examples of constraints are presented in a textual form and defined by means of OCL expressions

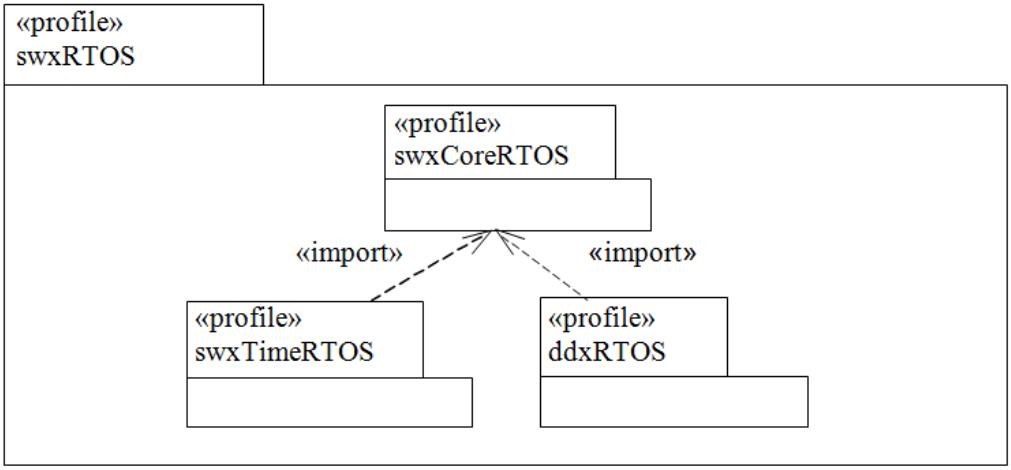
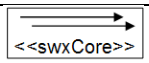
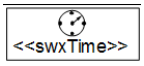


Fig. 1. Profiles for the creation of platform models

Table 1. Elements of the profiles of the platform models

	swxCore RTOS profile	
Stereotype	Description	Anotation
swxCore	Concepts regarding the software description in concurrent execution contexts. Extension of the *class* meta-class.	
Tagged values		
Name	Description	Type
pointer Reply Msg	Pointer at the memory address	Pointer
startAddressMsg	Start address of the outgoing message	Integer
thread ID	Thread Identifier	Integer
thead Name	Thread name	String
thread Priority	Thread priority	String
activateScheduler	Scheduler activation	Operation
activateThread	Thread creation	Operation
Send Put Msg	Send asynchronous messages	Operation
Sleep Thread for	Suspend the thread for a definite time	Operation
Constraints: The thread Name (tagged value) must have a unique name *Context swxCore-inv: self. thread Name select (is Stereotyped ('swxCore'))* *->forAll (s | s.thread Name = self. Thread Name implies s = self)* The thread Priority (tagged value) cannot be empty. *Context swxCore - inv: self. thread Priority select (is Stereotyped ('swxCore'))* *->for All (s|s.thread Priority = self. Thread Priority->not Empty( ))*		
Interaction point	Establishes the link between the application and the platform. Extension of the *dependency* meta-class.	
Tagged values		
Name	Description	Type
operationSource	Name of the source class operation defined in the PIM.	String
operationTarget	Name of the target class operation defined in the swxRTOS profile.	String
swxTimeRTOS Profile		
Stereotype	Description	Anotation
swxTime	Concepts regarding time values. Extension of the *class* meta-class.	
Tagged values		
Name	Description	Type
Nanosec	Store time related values in nanoseconds	Integer
Time	Store time related values.	Integer
ddxRTOS Profile		
Stereotype	Description	Anotation
ddxKeyboard	Concepts related of a keyboard	
Tagged values		
iniKBD	Initializes the keyboard	Operation
registerRec	Register a thread that will receive the key	Operation

Exemplification of UML Profile

In order to demonstrate the use of the AMP profile and the swxRTOS profile, an example of an alarm system is considered (Fig. 2). The purpose of this system is the simulation of an alarm and its basic functions such as alarm activation/deactivation and hardware device drivers control (keyboard, display and leds). The package named Alarm Application represents step 1, depicting the PIM model of this application and the use of the AMP profile. The AMP is used in order to abstractly capture concerns related to RTOS X Real-Time Kernel services. In this context, the "CCtrlMain" class represents the main class of the alarm system. It is composed of: "Init Thread Alarm" operation-responsible for creating and controlling the alarm thread; "Start" operation-starts up the RTOS, using the <<rtSwoperation>> stereotype of the AMP Profile to represent a software service; "Start Devices" operation-starts an RTOS device driver, using the <<rtD Doperation>> stereotype of the AMP Profile to represent a device driver service. In addition, the "CAlarm" is also a class of the alarm system, being responsible for controlling the main alarm functions.

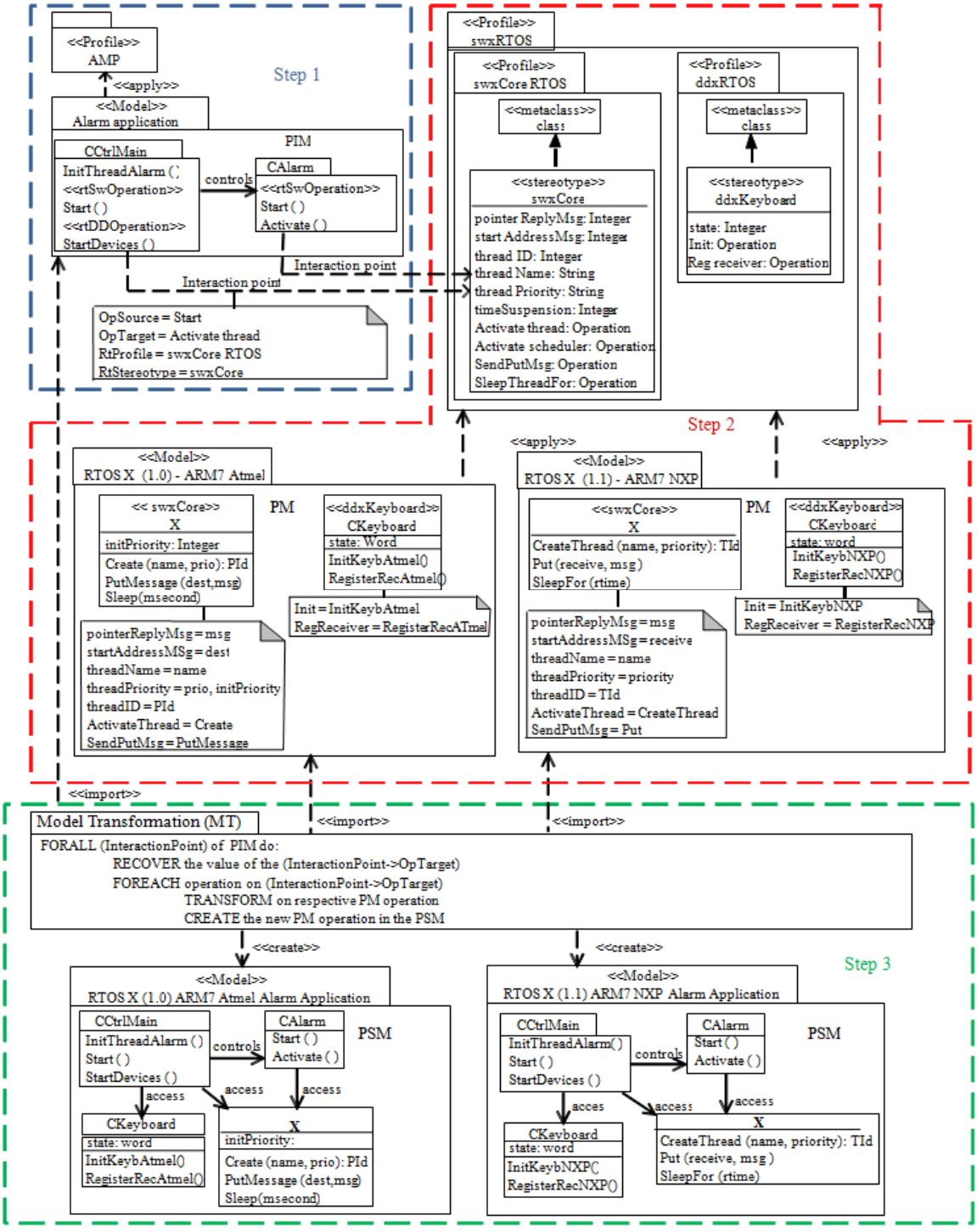


Fig. 2. Example of the Profiles for the creation of the Application and the Platform Models

In the example, only two operations are described: “Start”operation-starts the alarm thread, using the <<rtSwoperation>> stereotype of the AMP Profile to represent an RTOS software service; and “Activate” operation-activates the alarm.

Step1, illustrated in Fig. 2, shows the link between the PIM and the platform. In order to perform such link, the <<InteractionPoint>> stereotype was defined.

This stereotype is an extension of the UML Dependency metaclass, defined in the swxRTOS profile.

The <<InteractionPoint>> stereotype specifies the operation of the source class indicated in the PIM and the operation related to the target class indicated in the swxRTOS profile. For an application to access a platform service through an Operation Source, it must be bound to a corresponding Operation Target. In this example, the "Start" operation of the "CCtrlMain" class is linked to the "Activate Thread" meta-property of the swxCore class of the swxRTOS profile. The "Start" operation is tagged as Operation Source and the "Activate Thread" meta-property is tagged as Operation Target.

Step 2 is exemplified by the use of the swxRTOS profile and by the following packages: PM RTOS X (1.0)-ARM7 Atmel and PM RTOS X (2.0)-ARM7 NXP. These PMs differ in the RTOS version as well as in the associated hardware. A reduced form of the swxRTOS profile (Fig. 2) is used to represent an abstraction layer of the RTOS X Real-Time Kernel in a generic way, besides depicting two sub-profiles: swxCore RTOS and ddxRTOS. The first sub-profile represents the <<swxCore>> stereotype, responsible for the software description in concurrent execution contexts. In its turn, the second sub-profile represents the <<ddxKeyboard>> stereotype and the general concepts of a keyboard. For instance, the swxCore RTOS sub-profile includes the <<swxCore>> stereotype, which in its turn is composed of meta-properties such as: Thread Priority-indicates the thread priority; Activate Thread-creates a thread; and thread Name-indicates a thread name.

The application of the swxCore stereotype in the X class of the PM RTOS X (1.0)-ARM 7 Atmel allows the addition of semantics to the model elements of the X class. That is possible due to tagged values, as illustrated in the note associated with the X class in Fig. 2. Likewise, it is possible to notice that the Create operation in the X class represents the creation of a thread, while the name attribute represents the name of a thread and both the init Priority and the thread Priority attributes represent the priority of a thread.

Step 3, illustrated in Fig. 2 depicts the PIM and PM models attached as input parameters in the model transformation. Two PMs are illustrated so as to point out the differences between them and to show the practicability in their use, although only one of them is selected for the Model Transformation (MT) (step 3). The MT is succinctly illustrated in Fig. 2, aiming to represent the principle of the MT.

MT is a fundamental theme in MDE. Transformation between models can be defined as the translation of a model from a higher abstraction level to a lower abstraction level, based on a set of clearly defined rules (Sendall and Kozaczynski, 2003). The independence between transformation rules and platform features was achieved by means of PMs explicitly defined, enabling the creation of transformations that are reusable in several platforms. The MT defines the use of specific services of the RTOS, replacing the meta-properties defined in the swxRTOS profile with the properties defined in the PM. For example, considering that the PM RTOS X (1.0) ARM7 Atmel was selected as input PM of the model transformations, it is observed that the "Activate Thread", "thread Name" and "thread Priority" meta-properties defined in the "swxCore" class of the swxCore RTOS profile are replaced with the "Create", "name" and "prio" properties defined in the "X" class of the PM RTOS X (1.0) ARM7 Atmel. Replacements also occur for the other properties described in the classes of this PM. The result of the MT is the generation of a package named PSM-RTOS X (1.0) ARM7 Atmel Alarm Application.

Conclusion

Today, one of the main problems found in tools that support MDE is the fact that little attention is paid to questions related to the platform features in the software development trajectory. As a result, MDE tools are limited to certain platforms and PIM-into-PSM model transformation processes.

In order to achieve efficient, easily adaptable model transformation processes, the specification of independent platform features is necessary. Concerning RTOS-based embedded software development, the benefits in using this approach become even more evident due to both the inherent complexity of this kind of software and the existence of a wide variety of applicable platforms.

In this way, the main contributions of this paper are the two new UML 2.0 profiles proposed. The AMP profile enables the system to mark the PIM, indicating the RTOS services used. In addition, the swxRTOS profile was defined to be applied to the construction of PMs in different versions of the X Real-Time Kernel and in different associated hardware. This profile also performs the link between the application and the platform.

In future works, usage pattern description and behavioral modeling may be applied to the swxRTOS profile so as to obtain an accurate description of the execution platform.

Funding Information

The authors have no support or funding to report.

Author's Contributions

All authors equally contributed in this work.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Dube, M.R. and S.K. Dixit, 2012. Modeling theories and model transformation scenario for complex system development. Int. J. Comput. Applic., 38: 11-18. DOI: 10.5120/4618-6847

eSysTech, 2007. Embedded systems technology. eSysTech.

Jeon, S., J. Hong, I. Song and D. Bae, 2009. Developing platform specific model for MPSoC architecture from UML-based embedded software models. J. Syst. Software, 82: 1695-1709. DOI: 10.1016/j.jss.2009.04.043

Karsai, G., S. Neema and D. Sharp, 2008. Model-driven architecture for embedded software: A synopsis and an example. Sci. Comput. Programm., 73: 26-38. DOI: 10.1016/j.scico.2008.05.006

Kent, S., 2002. Model driven engineering. Proceedings of the 3rd International Conference on Integrated Formal Methods, May 15-18, Turku, Finland, pp: 286-298. DOI: 10.1007/3-540-47884-1_16

Kolovos, D.S, R.F. Paige and F. Polack, 2008. The Epsilon transformation language. Proceedings of the International Conference on Model Transformation, Zurich, Switzerland, Jul. 1-2, Zürich, Switzerland, pp: 46-60. DOI: 10.1007/978-3-540-69927-9_4

Kukkala, P., J. Riihimaki, M. Hamalainen and K. Kronlof, 2005. UML 2.0 profile for embedded system design. Proceedings of the Automation and Test in Europe Conference, Mar. 7-11, IEEE Xplore Press, pp: 710-715. DOI: 10.1109/DATE.2005.321

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

Maeng, J., J.H. Kim and M. Ryu, 2006. An RTOS API translator for model-driven embedded software development. Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Aug. 16-18, IEEE Xplore Press, Sydney, Qld., pp: 363-367. DOI: 10.1109/RTCSA.2006.15

MARTE, 2008. A UML profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, Beta 2 (convenience document without change bars). OMG Adopted Specification.

UML, 2011. OMG Unified Modeling Language (OMG UML), !Infrastructure, V2.1.2. Unified Modeling Language.

Renaux, D.P.B., R.E. Góes 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 (RES' 10), Gramado, Brazil.

Rosado, D.G., E. Fernández-Medina and J. López, 2011. Towards a UML extension of reusable secure use cases for mobile grid systems. IEICE Trans. Inform. Syst., E94-D: 243-254. DOI: 10.1587/transinf.E94.D.243

Selic, B., 2005. On software platforms, their modeling with UML 2 and platform-independent design. Proceedings of the of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, May 18-20, IEEE Xplore Press, pp: 15-21. DOI: 10.1109/ISORC.2005.40

Sendall, S. and W. Kozaczynski, 2003. Model transformation: The heart and soul of model-driven software development. IEEE Software, 20: 42-45. DOI: 10.1109/MS.2003.1231150

Truyen, F., 2006. The fast guide to model driven architecture-the basics of model driven architecture. Cephas Consulting Corp.

Wagelaar, D. and V. Jonckers, 2005. Explicit platform models for MDA. Proceedings of the ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Oct. 2-7, Montego Bay, Jamaica, pp: 367-381. DOI: 10.1007/11557432_27

Warmer, J.B. and A.G. Kleppe, 2003. The Object Constraint Language: Getting Your Models Ready for MDA. 2nd Edn., Addison-Wesley Professional, Boston, ISBN-10: 0321179366, pp: 206.