MT-AMP: A Model Transformation for Embedded Software

Luciane Telinski Wiedermann Agner[1*], Inali Wisniewski Soares[1], Paulo Cézar Stadzisz[2], Jean Marcelo Simão[2]
[1] Department of Computer Science, UNICENTRO - Mid-West State University, Guarapuava, Parana, Brazil.
[2] CPGEI, UTFPR - Federal University of Technology Paraná, Curitiba, Parana, Brazil.

* Corresponding author. Tel.: +55 42 36298344; email: lagner@unicentro.br.
Manuscript submitted November 16, 2015; accepted January 12, 2016.
doi: 10.17706/jsw.11.3.326-337

Abstract: MDA is an approach to software development based on the design and transformation of models. In the MDA approach, models are the core artifacts throughout the software development lifecycle, and thus a key role is played by model transformations. Nevertheless, most MDA initiatives are dedicated, i.e., the platform features are implicitly employed in the transformation rules. As the aspects associated with the platform are not separated from the transformation rules, for each selected platform there must be a corresponding model transformation. This makes the model transformation development easier, although the model transformation becomes limited to a specific platform, once the platform features are strongly associated with the transformation rules. An open challenge consists of how to create transformation rules independently of the platform features. This paper presents a model transformation, called MT-AMP, particularly applied to embedded software development based on PI-MT Method. In addition, it allows the use of the model transformation to different RTOS-based platforms by means of Platform Models explicitly defined. As a result, the MT-AMP provides independence between the model transformation rules and the platform features.

Key words: Model driven architecture, model transformation, platform model, real-time operating systems.

1. Introduction

Model Driven Architecture (MDA), a standard from the Object Management Group (OMG), is a software development approach based on model designing with high level of abstraction and platform independent [OMG]. MDA aims to make the application specification independent from the underlying platform or technology [1]. In order to do so, MDA provides means to separate platform aspects from implementation aspects by supporting the automatic transformation, from the modeling stage to the implementation stage.

Software development based on the MDA approach begins with the design of a Platform Independent Model (PIM). The conceptual design of software functionalities is captured in the form of a PIM, a model represented at a high abstraction level. The PIM model is transformed into a Platform Specific Model (PSM), based on a Platform Model (PM) [1]. A PM defines a set of concepts that represents a concrete platform [2]. Thus, PIMs should survive the advancements in technology and software architectures. In its turn, a PSM is also a model of software containing details of the target platform such as a specific operating system, a software architecture or an execution platform.

The transformation is an essential issue in the MDA approach, focusing on automated model transformations. In the MDA context, software development consists of a series of successive transformations of a model into another. Nevertheless, most MDA initiatives are dedicated, i.e., the platform

features are implicitly employed in the transformation rules [3]. As the aspects bound to the platform are not separated from the transformation rules, for each selected platform there must be a corresponding Model Transformation (MT) [4]. On the one hand, this makes the transformation development easier. On the other hand, model transformation becomes limited to a specific platform, once the platform features are strongly associated with the transformation rules. An open challenge consists of how to create platform independent model transformations.

This paper presents a model transformation, called MT-AMP, which provides independence between the transformation rules and the platform features as well as supporting the design and evolution of embedded software, particularly with respect to the use of RTOS. The MT-AMP is the refinement of a PIM by adding details regarding a particular target platform. Specific platform features are thus treated in an independent way by means of a Platform Model (PM) used as input in the transformation. The use of an explicitly defined PM enables the creation of transformations that are reusable in new platforms.The proposed transformation focuses on the static modeling and supports UML class diagrams.

The MT-AMP is based on the PIM-MT Method [11] and expressed in the Atlas Transformation Language (ATL), one of the most popular and widely used model transformation languages [5], [6]. In addition, it integrates some of the benefits from ATL superimposition to provide an efficient approach for the implementation of refinement in transformations with multiple input models [7].

This paper is organized as follows: Section 2 describe the RTOS-based platforms, Section 3 details the PI-MT Method, Section 4 presents the proposed model transformation, Section 5 shows an illustrative example demonstrating the application of this transformation, Section 6 brings the discussions and future works. Finally, conclusions are presented in Section 7.

2. RTOS-Based Model Platform

A platform is any set of hardware or software mechanisms that enable the execution of software applications. A Platform Model (PM) defines a set of concepts that represent a concrete platform [2]. Specifically, embedded software based on RTOS benefits from the use of an explicitly defined PM, mainly due to the wide variety of existing platforms. Such platforms, in their turn, are formed by a specific RTOS and an associated processor. The PM must specify the set of hardware and software mechanisms that enable the execution of embedded software applications. The software refers to the RTOS and its respective APIs, while the hardware refers to the platforms based on specific processors. Depending on the platform employed by the application, a specific PM must be selected and used as execution input model of the models transformation.

The input platform model of the model transformations implemented in the proposed transformation is defined through the application of elements from a platform profile, called swxRTOS [10]. This profile takes the X Real-Time Kernel RTOS as part of the target platform [8]. This RTOS is structured in modules, including: microkernel, hardware abstraction layer (X-HAL), shell, event tracing, TCP/IP stack, and USB stack. The X Real-Time Kernel can be employed in different hardware architectures, identified according to the processor used: ARM7TDMI, ARM9, Cortex-M4, among others. This way, several platforms can derive from this RTOS. The swxRTOS profile provides a practical way for promoting independence in MDA transformations by means of UML stereotypes that correspond to the X RTOS services. A UML stereotype defines how an existing metaclass may be extended and enables the use of platform in place of the ones used for the extended metaclass [9].

2.1. swxRTOS Profile

The swxRTOS profile is part of the PROAPES profile and defines a set of stereotypes in order to describe the services provided by a platform that makes use of the RTOS and their respective hardware platforms of

embedded systems, based on specific processors [10]. The swxRTOS is used to describe PMs in structural models (i.e., class diagram). This profile, briefly illustrated in Fig. 1, is composed of the following subprofiles:

- swxCoreRTOS: represents the basic concepts of high level designing, required to support the use of concurrent designing and iterations.
- swxTimeRTOS: represents the basic time-related concepts.
- swxShellRTOS: represents the depuration supporting concepts, which allow the user to track the processor resource allocation and use by its application.
- ddxRTOS: consists in providing modeling artifacts for the description of device drivers employed in an embedded system and that make use of the RTOS X Real-Time Kernel.

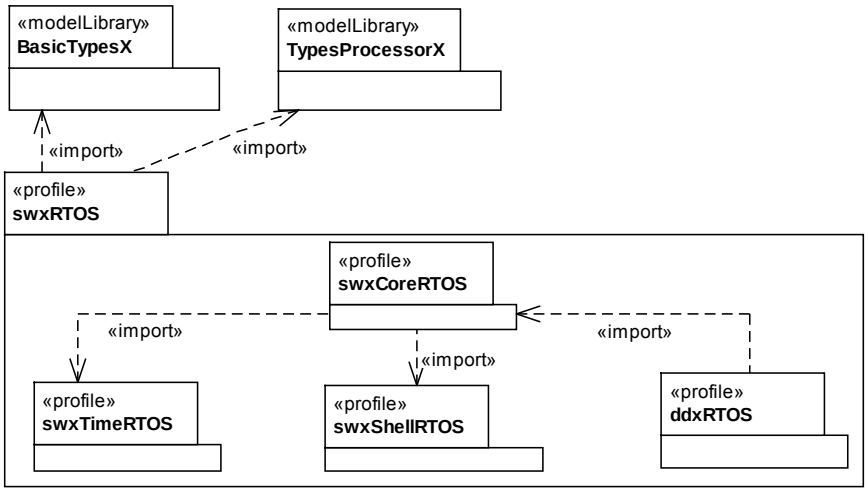

Fig. 1. swxRTOS profile.

The swxRTOS subprofiles import the data types defined in the BasicTypesX model library as well as the process types defined in the TypesProcessorX model library. The swxCoreRTOS subprofile regards the main concepts of an operating system, and thus it is one of the most used. This subprofile defines five stereotypes, as follows:

- swxCore: represents the basic concepts for context description of software concurrent execution for the RTOS X Real-Time Kernel.
- swxSemaphore: represents the basic concepts for the creation and management of a semaphore for the RTOS X Real-Time Kernel. In this way, tasks can be mutually synchronized by implementing the RTOS X Real-Time Kernel critical regions.
- swxPipe: represents a pipe between two tasks by storing the references to the data objects.
- swxISR_Pipe: represents a pipe between an anonymous ISR and a thread that stores the references to the data objects.

The platform profile is used in the design of Platform Models, proving means to obtain platform independence for a group of embedded computational platforms. Additionally, a linking profile defines stereotypes to be employed in PIM model elements so as to specify which of these elements make use of the implementation platform services. The linking profile thus enables the annotation of the application model elements that will employ the platform services, as defined in the PM. The transformation proposed in this paper uses the linking profile AMP, as presented next.

2.2. AMP Profile

The Application Modeling Profile (AMP) linking profile is used by the MT-AMP transformation to

annotate PIM models. This profile defines a set of stereotypes that can be employed to annotate operations of a PIM model, represented by a class diagram. Each stereotype represents a set of RTOS X Real-Time Kernel services. Therefore, the stereotypes defined in the AMP allow the design of a PIM with a sufficient platform independence level, permitting its transformation into specific platform models according to the selected RTOS and the respective associated hardware.

Fig. 2 shows part of the AMP profile and presents the ServiceRTOS stereotype, defined through the extension of the Operation metaclass. This stereotype defines the following properties (tagged-values):

- rtService - stores the name of the stereotype to be searched in the platform model;
- opTarget - stores the name of the RTOS operations used by the PIM elements.

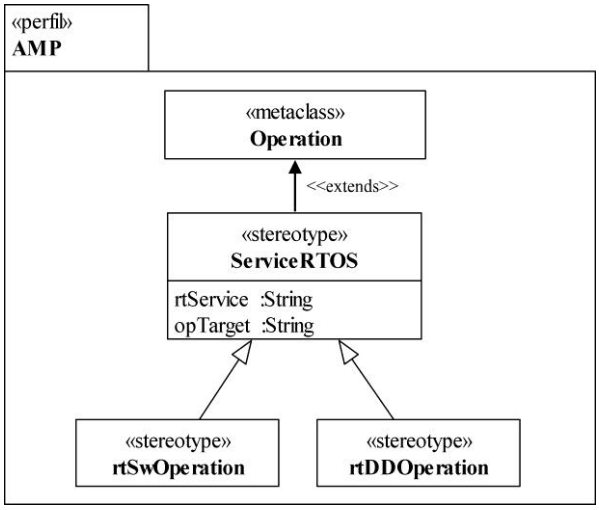

Fig. 2. AMP profile.

The rtSwOperation and rtDDOperation stereotypes generalize the ServiceRTOS stereotype and are used to annotate the PIM elements in order to identify the RTOS services abstractly employed by the application. In this way, an RTOS service can be a software operation service for an RTOS defined through the rtSwOperation stereotype, or a device driver operation service defined through the rtDDOperation stereotype.

3. PI-MT Method

The PI-MT (Platform Independent - Model Transformations) is a method that aims at the development of platform independent model transformations in the context of RTOS-based software development. This method benefits from enabling the adaptation of the model transformation process to different platforms by means of Platform Models explicitly defined. As a result, the PI-MT method provides independence between the model transformation rules and the platform features [11].

The main steps to be performed for the use of the PI-MT method, illustrated in Fig. 3, are the following: 1) Transformation Model Creation; 2) PIM Definition; 3) PM Selection; and 4) Transformation Execution [11]. The PI-MT method makes use of Platform Models explicitly defined and aims to systemize the process of creating adaptable Model Transformations.

4. The Mt-Amp Transformation

The MT-AMP model transformation, deriving from the PI-MT method, aims to provide platform independence in PIM-to-PSM model transformations, focusing on the development of RTOS embedded software. The MT-AMP focuses on the static modeling of the software system and supports the employment of UML class diagrams.

The proposed transformation framework is presented in Fig. 4. In this figure, the colored boxes represent the transformation flow by means of the input models (PM and PIM), output model (PSM), and transformation models (PIM2PSM.atl and UML2Copy.atl). The other boxes represent the metamodels used in the definition of such models (UML metamodel, ATL metamodel, MOF, swxRTOS profile, and AMP profile).

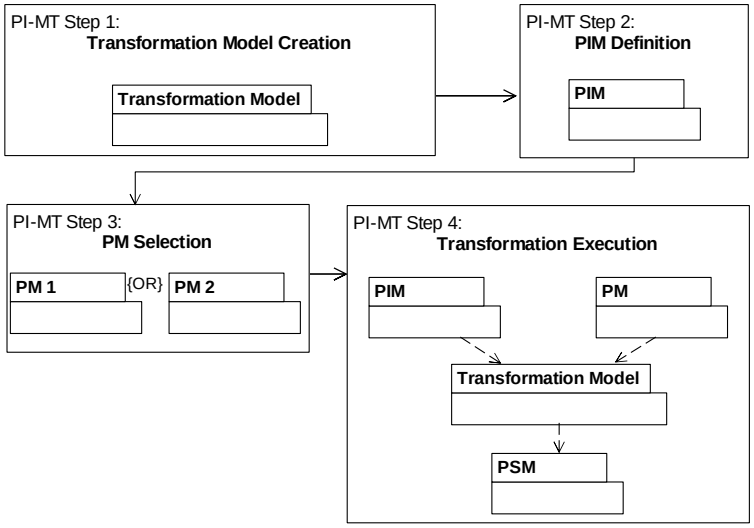

Fig. 3. Main steps of the PI-MT method.

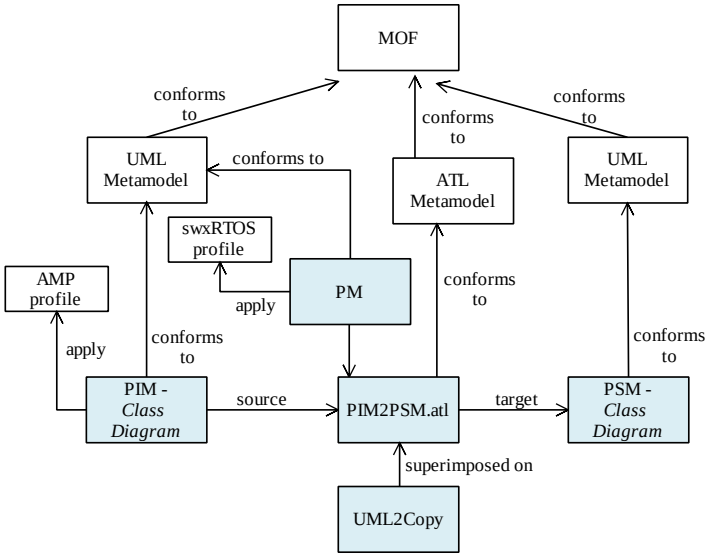

Fig. 4. MT-AMP transformation.

The MT-AMP is driven by the transformation module called PIM2PSM.atl, which contains the specific rules that perform the changes in the source model elements related to the RTOS services. The PIM2PSM.atl module superimposes the UML2Copy.atl module through the module superimposition technique [7]. The UML2Copy.atl module, proposed and made public by Wagelaar *et al.* (2010), copies all elements of an input UML model to an output model [7]. For each UML element of the input model, the UML2Copy.atl module copies such element to the output model.

4.1. MT-AMP Definition

The MT-AMP transformation was implemented in ATL, a hybrid transformation language dedicated to model transformation, i.e., containing declarative and imperative constructs [12]. Hybrid approaches allow

developing more complex applications [13]. The use of ATL language is appropriate to the development of this model transformation, once it provides mechanisms to produce a target model from a set of source models. Also, it can express model transformations in the MDA context based on explicit metamodel specifications [12]. In this way, rules are able to clearly state how concepts from source metamodels are mapped to concepts from the target ones.

The target platform of this transformation is based on the RTOS X Real-Time Kernel employed in ARM processors [8]. The MT-AMP must replace RTOS generic services defined in the PIM by platform-specific operations (indicated in the PM). The transformation model must also define a PIM-PM interface, seeking to identify how the PIM elements must be related to the PM elements to generate a PSM model with features of a specific platform.

In order to establish such link, the transformation makes use of rtSwOperation and rtDDOperation stereotypes defined in the AMP profile. These stereotypes must be applied to the PIM model elements so as to identify the elements that abstractly employ the RTOS software and device driver services.

The MT rules must search the PM for elements annotated in the PIM and, then, create them in the PSM. Fig. 5 presents these steps, performed by the MT-AMP transformation model and succinctly described in natural language.

- **FOR EACH** existing operation in the PIM model:
- **IF** the operation is annotated with the <<rtSwOperation>> sterotype **THEN**
  - Recover the value of the rtService property.
  - Search the PM for the class that contains the stereotype whose name matches the value of the rtService property related to an RTOS software service.
  - Create a new class in the PSM model, with the same name of the PM found.
  - Recover the operations defined in the opTarget property and create a respective operation in the PSM for each existing operation.
  - Create a link between the new PSM class and the corresponding PIM class.
- **ELSE**
  - **IF** the operation is annotated with the <<rtDDOperation>> stereotype **THEN**
    - Recover the value of the rtService property.
    - Search the PM for the class containing the stereotype whose name matches the value of the rtService property related with an RTOS device driver service.
    - Create a new class in the PSM model with the same name of the PM class found.
    - Recover the operations defined in the opTarget property and create a corresponding operation in the PSM for each existing operation.
    - Create a link between the new PSM class and the corresponding PIM class.
  - **ELSE**
    - Make a copy of such operation.

Fig. 5. MT-AMP steps.

4.2. MT-AMP Implementation

The proposed transformation is a refinement expressed by endogenous transformations. In refinement, most elements must simply be copied, while others must be changed. As a result, the transformation preserves most parts of the source model [14]. Besides, endogenous transformations are transformations between models expressed in the same metamodel.

The proposed solution uses UML-based metamodeling and refers to the UML metamodels. PIM, PM, and PSM thus conform to the UML metamodel. Currently, the UML metamodel is widely used and it is supported by several development tools [15]. Also, this work integrates some of the benefits from ATL superimposition to provide an efficient approach for the implementation of model refinement [15].

Superimposition allows the composition of two or more transformation modules in one single transformation execution by extending and overriding rules in transformation modules. Also, the superimposition enables the separation of the copying common rules from the refinement specific rules [16]. Consequently, reusability is improved by the ability to extend and adapt generic transformation modules. In this work, the transformation rules of the UML2Copy.atl module, proposed by Wagelaar *et al.* [7], are reused and overridden by the PIM2PSM.atl transformation module, when necessary.

The UML2Copy.atl module provides the copy of a UML model and can be used as base for implementing refinement transformations. This module is based on the UML metamodel, and includes a transformation rule for every metaclass from which it must copy the instances. By separating the general copying functionality (defined in the UML2Copy.atl module) from the specific details of the transformation (defined in the PIM2PSM.atl module), it is possible to improve maintainability, since it becomes easier to find and solve specific problems.

4.3. PIM2PSM.atl Module

The PIM2PSM.atl module has multiple input models, namely: the PIM source model and the Platform Model. The output model refers to the PSM and is created as a result of the transformation. ATL supports the definition of custom operations by means of helpers. A helper aims to perform navigation over the source models. Once the model transformation presented in this paper is founded on a UML profile and considering the main impacts that stereotypes and property values have on this implementation, it is important: 1) to check if a certain stereotype is applied to a model element; and 2) to query the value of a certain property value definition in a specific stereotype. As a consequence, the PIM2PSM.atl module contains several helpers (described in Table 1) that determine whether a specified stereotype is applied to an element, and retrieve the value of a property from a specified stereotype.

Table 1. PIM2PSM.atl Helpers

***Helper* Name**	***Helper* Description**
getTagVal	Returns a property value (*tagged value*) associated with a specific stereotype.
getPMClass	Searches the PM for a class containing a given stereotype applied. Gets the name of the stereotype from the *swxRTOS* subprofile to be found in a PM class as parameter.
isStereotype	Checks if a source-model Operation has an applied stereotype.
isrtSwOperation	Checks if a source-model Operation has an *rtSwOperation* stereotype applied.
isrtDDOperation	Checks if a source-model Operation has an *rtDDOperation* stereotype applied.

In the scope of the ATL language, the generation of target model elements is achieved through the specification of transformation rules. The PIM2PSM.atl rules search in the PIM for annotated operations that contain specific stereotypes, create new classes and their corresponding RTOS operations in the PSM, and create the respective association between the existing PIM classes and the new classes generated in the PSM. Table 2 shows the rules defined in the PIM2PSM.atl module.

Table 2. PIM2PSM.atl Rules

Rule Name	***Rule Description***
Operation	Copies the non-annotated operations of the PIM model to the PSM model.
OperationStereotype	Searches the source-model operations annotated with a stereotype.
OperationRtSw	Searches the source-model operations annotated with a <<rtSwOperation>> stereotype and based on the stereotype property information performs specific tasks to insert platform features regarding RTOS software services in the PSM.
OperationRtDD	Searches the source-model operations annotated with a <<rtDDOperationt >> stereotype and based on the stereotype property information performs specific tasks to insert platform features regarding RTOS device driver services in the PSM.
Model	Copies the PIM model properties to the PSM model.

5. An Illustrative Example

In order to illustrate the proposed transformation, a simplified example is shown in Fig. 6. The example performs the transformation based on the PM RTOS X - eAt55, which considers the X Real-Time Kernel version 1.0 in ARM7 processors. The diagrams outlined in this example represent a system responsible for showing messages on the display. A class diagram presents a PIM model fragment regarding the messaging control module of an application. In this example, the PIM model contains two classes: “CtrlMsg” and “CDisplay”. The “CtrlMsg” class controls the flow of messages sent or received by RTOS processes under execution.

In the example, only the “sendMsg” operation is defined in the “CtrlMsg” class, so it is responsible for sending messages to RTOS processes under execution. In its turn, the “CDisplay” class controls a monitor connected to the ARM7 processor and, in this brief example, defines only the “clear” operation, responsible for removing messages from the display, and the “displayMsg” operation, responsible for showing messages on the display.

The swxRTOS profile, briefly illustrated beside the PIM model, is used to represent an abstraction layer (platform profile) for the RTOS X Real-Time Kernel version 1.0.The link between the PIM and the platform is obtained through interaction points, which correspond to the rtSwOperation and rtDDOperation stereotypes.

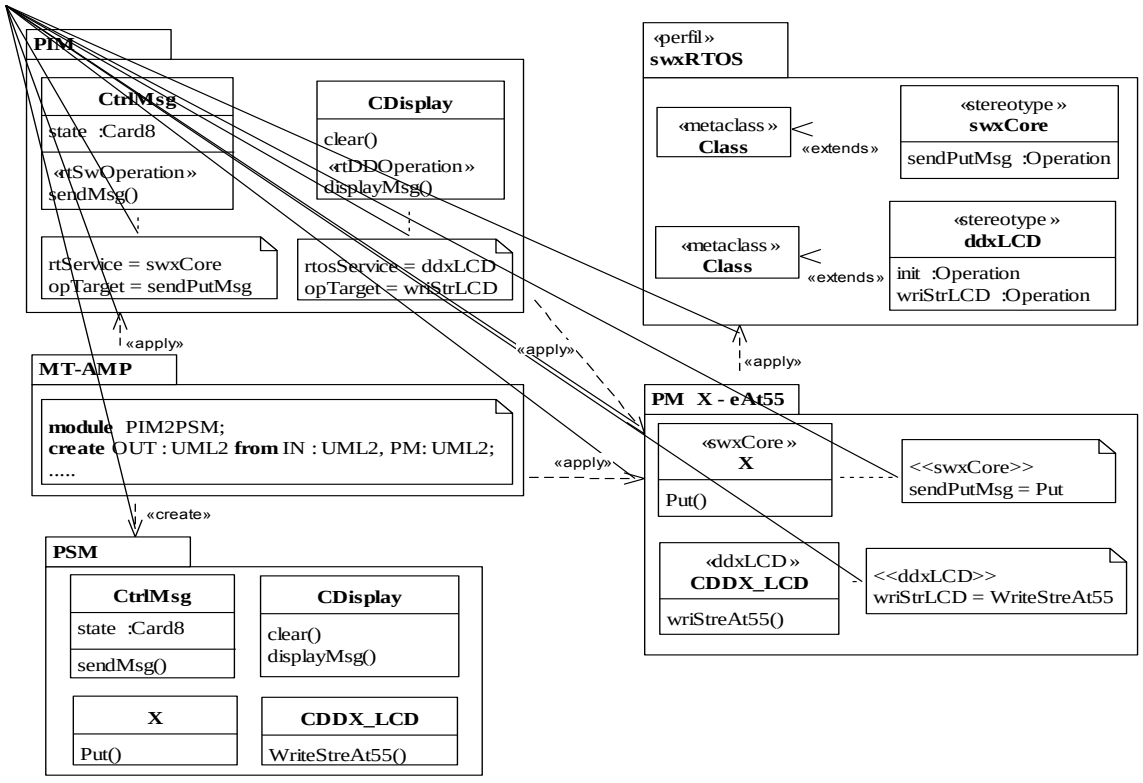

Fig. 6. Example of the PIM into PSM transformation.

In this way, the transformation searches the PIM model for elements annotated by those stereotypes and, based on information defined by properties (tagged values), searches the PM for the elements to be inserted in corresponding target model (PSM). A new class is thus created in the PSM to provide the services indicated in the PIM. The “sendMsg” operation of the “CtrlMsg” class is linked to the “sendPutMsg”

operation of the swxRTOS profile by tagged values associated with the rtSwOperation stereotype.

The rtSwOperation stereotype defines the following properties: rtService and opTarget. The rtService indicates the stereotype to be searched in the PM model, in this case the swxCore stereotype. In its turn, the opTarget stores the RTOS services indicated in the PIM according to elements of the swxRTOS profile, in this case the "sendPutMsg" operation. Likewise, the "displayMsg" operation annotated with the rtDDOperation stereotype is defined in the PIM "CDisplay" class and associated with the "wriStrLCD" RTOS service through the opTarget property.

The transformation, succinctly illustrated in Fig. 6, is based on the PM - eAt55: RTOS X Real-Time Kernel (version 1.0) coupled with an ARM7 processor. In this way, the transformation instance shows the generation of a PSM model based on this reference platform. The swxCore stereotype (defined in the swxRTOS profile) is applied to the "X" class of the PM as well as the ddxLCD stereotype is applied to the PM "CDDX_LCD" class. By means of property values (tagged values), the transformation rules substitute the operations defined in the swxRTOS profile for the corresponding operations defined in the PM.

6. Discussion and Related Work

The high demand for new embedded products with additional functionalities is a trend, thus requiring the increment of more complex software components. As a result, the increasing complexity of embedded software systems emphasizes the need for development approaches that provide better productivity and quality, e.g. the MDA [17].

In the same way, embedded software is commonly subject to rigid restrictions, once it is developed for a specific platform, i.e., a particular combination of basic hardware and software. That impairs the reuse of the developed software in different platforms, given the required customization according to the adopted platform. Such factors represent a big challenge for the embedded software development community [18]. According to Espinoza *et al.* (2009), the use of model driven approaches under the concept of embedded software promotes the software reuse and evolution [19].

Given the evolution of embedded systems, with additional functions and higher complexity, the use of embedded operating systems has constantly increased. Such systems are called Real-Time Operating Systems – RTOS and provide a set of standardized services for the management of the embedded hardware resources. An RTOS differs from a common-use operating system mainly because it supports the execution of embedded systems [20]. The support provided by MDA for the development of embedded software, manly RTOS-based embedded software, is still limited.

Model transformation plays a key role in MDA. Most researches on model transformation, in the MDA context, are dedicated and define the platform aspects together with transformation rules [21]-[24]. In these cases, an explicitly defined Platform Model (PM) is not used and the transformation becomes restrict to a specific platform [3], [13]. Some initiatives adopt the concept of explicitly defined PM,[2], [4], [25], [26]. Such initiatives aim at the design and use of Platform Models, although they do not provide specific artifacts for modeling RTOS-based embedded systems services.

In model transformations that do not make use of an explicitly defined PM, there must be a model transformation configured for each target platform employed. The MT-AMP's highlight regards the creation enabling of generic transformations, applicable to new platforms, through the use of explicitly defined PMs and the definition of transformation rules regardless the adopted platform.

A framework for model transformations based on the Modeling and Analysis of Real-Time and Embedded Systems (MARTE) profile is proposed by Chehade *et al.* (2011) and focuses on the conception of generic model transformations oriented to applications based on Real-Time Operating Systems [27].

The MARTE profile defines elements that enable the modeling of real-time embedded systems in different

RTOSs [28]. It is important to point out that RTOS-based embedded platforms have APIs (Application Program Interfaces) and implementation patterns that can vary considerably, requiring a very wide range to be covered by the MARTE profile. Therefore, the software system modeling for a specific RTOS using MARTE requires the adaptation of this profile to the modeling conventions of the selected operating system as well as deep knowledge of the developer on the elements defined in this profile. In this case, the main problems found regard the MARTE complexity and generality of use, which can be adapted to different RTOS and comprises several elements in its specification.

The MT-PROAPES transformation also provides independence between the transformation rules and the platform features by means of the PROAPES profile [10]. However, the MT-PROAPES is based on dyxRTOS sub-profile and supports only behavioral models (sequence and activity diagrams). In its turn, the MT-AMP transformation is based on swxRTOS profile and supports class diagrams.

In future works the MT-AMP may be adapted to other RTOS based platforms. Also, the proposed transformation was implemented in ATL language. Thus, the use of another model transformation language may be subject of study in future works.

7. Conclusion

The main contribution of this paper lies in proposing a model transformation that provides independence at different levels of embedded software development, from the modeling to the transformation stage. In this way, the same model transformation rules can be used in different platforms.

For that, a PM containing specific features of a particular platform must be selected, once such platforms consist of an RTOS linked to embedded hardware platforms and based on specific processors. Explicitly defined PMs can bring significant contributions to model transformation development in terms of independence enhancing between models and platforms, thereby enabling the reuse and increasing productivity, particularly in RTOS-based embedded software development.

Therefore, this research paper's highlight consists in providing "platform independence" both at the application modeling and at the model transformation level. As a matter of fact, dedicated model transformations merely transfer the "platform independence" issue from the application development to the model transformation development. In embedded systems, this issue is even more critical, once such systems cab be implemented through the use of several platforms, each of them with specific characteristics and restrictions.

References

[1] Object management group - OMG. (2003). *MDA Guide Version 1.0.1*. Retrieved November 2015, from http://www.omg.org/cgi-bin/doc?omg/03-06-01

[2] 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* (pp. 15 - 21)

[3] Tratt, L. (2005). Model transformations and tool integration. *Software and Systems Modeling*, *4(2)*, 112 - 122.

[4] Wagelaar, D., & Jonckers, V. (2005). Explicit platform models for MDA. *Proceedings of the ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems*.

[5] Sanchez, C. J., Guerra, E., & De, L. J. (2014). Uncovering errors in ATL model transformations using static analysis and constraint solving. *Proceedings of the IEEE 25th International Symposium on Software Reliability Engineering* (pp. 34 - 44).

[6] Troya, J., & Vallecillo, A. (2011). A rewriting logic semantics for ATL. *Journal of Object Technology*, *10(5)*,

1 - 29.

[7] Wagelaar, D., Van, Der, S. R., & Deridder, D. (2010). Module superimposition: a composition technique for rule-based model transformation languages. *Software and Systems Modeling*, *9(3)*, 285 - 309.

[8] Renaux, D. P. B., Goes, R. E., & Linhares, R. R. (2010). Performance characterization of real-time operating systems for systems-on-silicon. *Proceedings of the 12th Brazilian Workshop on Real-Time and Embedded Systems* (pp. 63 - 74).

[9] Object management group - OMG. (2011). *Unified Modeling Language (UML): Superstructure*. Retrieved November 2015, from http://www.omg.org/spec/UML/2.4/Superstructure/Beta2/PDF

[10] Soares, I. W., Agner, L. T. W., Stadsizs, P. C., & Simao, J. M. (2012). Modeling of embedded software on MDA platform models. *Journal of Computer Science and Technology*, *12(3)*, 133 - 139.

[11] Agner, L. T. W., Soares, I. W., Stadsizs, P. C., & Simao, J. M. (2012). PI-MT: A method for the creation of generic model transformations. *Proceedings of the 21st International Conference on Software Engineering and Data Engineering* (SEDE).

[12] Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I. (2008). ATL: A model transformation tool. *Science of Computer Programming*, *72(1 – 2)*, 31 – 39.

[13] Czarnecki, K., & Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, *45(3)*, 621 – 645.

[14] Kurtev, I. (2008). State of the art of QVT: A model transformation language standard. *Proceedings of the International Workshop on Applications of Graph Transformation with Industrial Relevance*.

[15] France, R., & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. *Proceedings of the Future of Software Engineering* (pp. 37 – 54).

[16] Wagelaar, D. (2008). Composition techniques for rule-based model transformation languages. *Proceedings of the International Conference on Model Transformation*.

[17] Karsai, G., Neema, S., & Sharp, D. (2008). Model-driven architecture for embedded software: A synopsis and an example. *Science of Computer Programming*, *73(1)*, 26 - 38.

[18] Marwedel, P. (2006). *Embedded System Design*. NJ, USA: Springer-Verlag New York.

[19] Espinoza, H., Selic, B., Cancila, D., & Gérard, S. (2009). Challenges in combining SysML and MARTE for model-based design of embedded systems. *Proceedings of the Fifth European Conference on Model Driven-Architecture Foundations and Applications*.

[20] Douglass, B. P. (1999). *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*.

[21] Jeon, S., Hong, J., Song, I., & Bae, D. (2009). Developing platform specific model for MPSoC architecture from UML-based embedded software models. *The Journal of Systems and Software*, *82(10)*, 1695 - 1708.

[22] Anastasakis, K., Bordbar, B., Georg, G., & Ray, I. (2010). On challenges of model transformation from UML to alloy. *Software and System Modeling*, *9(1)*, 69 - 86.

[23] Sun, Y., White, J., & Gray, J. (2009). Model transformation by demonstration. *Proceedings of the 12th Model Driven Engineering Languages and Systems*. Denver.

[24] Lopes, D., Hammoudi, S., Bézivin, J., & Jouault, F. (2005). Mapping specification in MDA: From theory to practice. *Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications*.

[25] Sandrieser, M., Benkner, S., & Pllana, S. (2011). Explicit platform descriptions for heterogeneous many-core architectures. *Proceedings of the 16th International Workshop on High-Level Parallel Programming Models and Supportive Environments*.

[26] Kukkala, P., Riihimaki, J., Hannikainen, M., Hamalainen, T. D., & Kronlof, K. (2005). UML 2.0 profile for embedded system design. *Proceedings of Conference on Automation and Test in Europe Conference*.

[27] Chehade, W. E. H., Radermacher, A., Gerard, S., & Terrier, F. (2010). Detailed real-time Software platform modeling. *Proceedings of the 17th Asia Pacific Software Engineering Conference*.

[28] Gerard, S., Petriu, D., & Medina, J. (2007). MARTE: A new standard for modeling and analysis of real-time and embedded systems. *Proceedings of the 19th Euromicro Conference on Real-Time Systems.*

Luciane Telinski Wiedermann Agner obtained the B.Sc. degree in computer science from Pontifical Catholic University of Paraná (PUC-PR) in 1991. In 2000, she received the M.Sc. degree from the Federal University of Paraná (UFPR). She obtained her PhD from Graduate Program in electrical engineering and industrial computer science at the Federal University in Technology of Paraná (UTFPR), Brazil, in 2012. She is currently a professor at the Department of Computer Science of the Mid-West State University (UNICENTRO), Brazil. Her publications and researches interests include software/system engineering and computer science.

Inali Wisniewski Soares obtained the B.Sc. degree in computer science from UEPG, Brazil, in 1990, and in 2000, she received her M.Sc. degree from the Federal University of Paraná (UFPR), Brazil. She received her PhD degree from Graduate Program in electrical engineering and industrial computer science (CPGEI) at the Federal University in Technology of Paraná (UTFPR), Brazil, in 2012. At present, she is a professor at the Department of Computer Science (DECOMP) of the Mid-West State University (UNICENTRO), Brazil. Her publications and researches interests include software/system engineering and computer science.

Paulo Cézar Stadzisz obtained the diploma in data processing technologies from the Federal University of Paraná, Brazil, in 1987. He received his M.Sc. degree in industrial computer science from the Graduate Program in Electrical Engineering and Industrial Computer Science (CPGEI) of the Federal Center of Technological Education, Brazil in 1990. He received the Ph.D. degree from the Franche-Comté University (France) in 1997. He is currently a professor at the Paraná Federal University of Technology (UTFPR). He teaches and coordinates researches in the CPGEI/UTFPR doctoral program. He coordinates the Laboratory of Innovation in Technology at UTFPR. His publications and researches include system modeling and analysis, simulation, and software engineering.

Jean M. Simão obtained his B.Sc. degree in computer science from UEPG in 1998 and in 2001 the M.Sc. degree from CPGEI/CEFET-PR/UTFPR, Brazil. In 2005, he obtained the Ph.D. degree in computing and automatic domains from CPGEI/UTFPR and the Research Center for Automatic Control of Nancy (CRAN) – Henry Poincaré University (UHP) France. In 2006 and 2014, he developed post-doctoral activities at CR. Nowadays, he is a professor at UTFPR in Departments (Electronics, Informatics-PPGCA, CPGEI) and Laboratories (Innovation in Technology, Intelligent Systems). His teaching activities concern computer science and his researches/publications include intelligent systems and programming paradigms.