UML Modeling of a Real-Time Embedded Industrial System: an Engine Testbed

Nicolas Pernet, Sylvie Cauvin, Frédéric Thomas, Chokri Mraidha, Michel Sall, Philippe Robin

UML Modeling of a Real-Time Embedded Industrial System: an Engine Testbed

Nicolas PERNET, Sylvie CAUVIN[1], Frédéric THOMAS, Chokri MRAIDHA [2], Michel SALL, Philippe ROBIN[3]

1: IFP, 1-4 avenue du Bois Préau, F-92852 Rueil Malmaison, France
2: CEA LIST, CEA Saclay, F-91191 Gif sur Yvette, France
3: TRIALOG, 25 rue du Général Foy, F-75008 Paris, France

Abstract: The paper presents a work performed in the Software Factory / OpenDevFactory project which is part of the French System@tic cluster. The OpenDevFactory project focuses on the modeling of complex systems using a wide variety of tools and technologies and the Eclipse framework as an integration platform.

The paper presents the work performed in an industrial use case proposed by French IFP research institute which, among other disciplines, is a specialist of internal combustion engine modeling, design and testing. The modeled system is a real-time distributed control system for engine testbeds.

Real-time and embedded systems dedicated UML modeling components developed within the OpenDevFactory project are used and combined to describe the deployment of the testbed control applications onto the underlying execution infrastructure based on RTAI real-time operating system. Modeling components are relevant to the modeling of the real-time and communication requirements of applications, the modeling of fault-tolerant requirements and the modeling of the underlying technical architecture. The deployment is performed down to the actual code using automatic code generation.

The paper presents the modeling process used in the industrial use case and its usability and will conclude on the portability and productivity enhancements achieved with this modeling process.

Keywords: Eclipse, Platform Independent Model (PIM), Platform Description Model (PDM), Platform Specific Model (PSM), UML modeling, real-time embedded, RTAI

1. Introduction

Model-based engineering currently offers new solutions to deal with real-time systems development. The recently adopted OMG MARTE[1] (Modeling Analysis of Real-time and Embedded systems) standard [1] brings improvements in three directions: enrichment / refinement of system functionalities, reduction of time-to-market and production costs, and compliance with non-functional requirements. Based on MARTE standard, CEA LIST developed the two components called CT-RTE and CT-PDM for the Eclipse framework. Another component named CG-LA was provided as an Action Language editor. At last, by combining these components with the new UML modeler named Papyrus[2], CEA LIST offers a comprehensive integrated platform for the modeling, analysis and synthesis of real-time and embedded systems.

IFP proposed an industrial use case which consists in the modeling of a real-time test-bed supervisor and used the new integrated platform proposed by CEA LIST for this purpose. In order to assess a complete model-to-code process, the project decided to target an implementation of the supervisor on the RTAI real-time operating system. In order to achieve the required model-to-code transformation, TRIALOG used the CT-PDM component to describe the RTAI software platform and developed an automatic code generation solution.

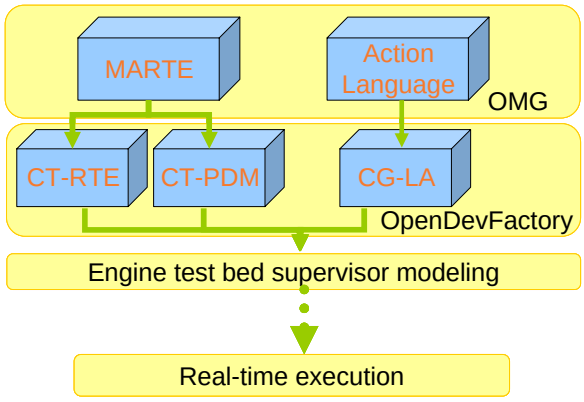

Figure 1: Use case principles and components foundations

This work was performed in the context of the Usine Logicielle project of the System@tic Paris Région Cluster. Figure 1 presents the different components used for the use case and the foundation of each of them.

[1] http://www.omgmarte.org/

[2] http://www.papyrusuml.org

2. Presentation of engine testbed use case

IFP, along with D2T company, has developed a unique solution for test-bed automation, engine calibration test implementation and model integration into the test-bed (simulation). The solution is based on a real-time control system working with RTX real-time kernel. Figure 2 shows a screenshot of the commercial version of the software.

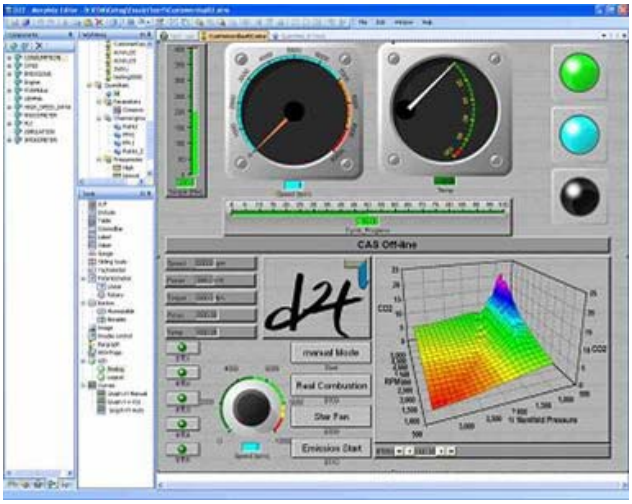

Figure 2: Real-time supervisor screenshot

In the OpenDevFactory project, IFP aimed at testing a new modeling process based on components dedicated to real-time systems. First, CT-RTE was used to specify non-functional properties such as real-time constraints or time behavior. Second, CG-LA offered a new solution to describe operations and algorithms, without choosing either a specific programming language or activity diagrams. Third, CT-PDM allowed IFP designers to specify the final implementation of the system. IFP chose to investigate an implementation based on the RTAI platform.

3. Assessment of modeling components

3.1 CT-RTE component

Description of component

CT-RTE stands for Technical Component for Real-Time and Embedded applications modeling.

The modeling of real-time applications requires a modeling language that enables the description of the specific features inherent to the real-time domain. Actually real-time applications have qualitative features such as deadline and period, as well as quantitative features related to communication, concurrency and behavioral aspects. The modeling language used to model real-time applications must at least provide the modeling concepts to represent such features.

UML [2] is a general purpose modeling language that provides concepts for behavioral modeling, some concepts for concurrency modeling (e.g. CallConcurrencyKind in Communications, concurrent states in StateMachines), and some basic concepts on time modeling. However, in order to guarantee that UML is a general purpose language, those concepts are not defined precisely enough to capture the specificities of a given domain (e.g. the real-time domain). To handle domain specificities, UML provides a specialization mechanism called *profiles*. Profiles provide the capability to tailor the UML metamodel for different domains. This is achieved by extending the UML metamodel elements in order to adapt them for the targeted domain. The UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) is a specialization of UML that provides concepts dedicated to the real-time domain.

CT-RTE component is based on the UML profile for MARTE. CT-RTE aims at providing a way to define and model the real-time characteristics of real-time and embedded systems, such as non-functional properties (NFPs MARTE sub-profile), time structure (Time MARTE sub-profile) (e.g. not-timed / asynchronous, synchronous, multi-clock). CT-RTE component allows designers to specify real-time quantitative features (e.g. deadline, energy consumption, memory footprint, etc), as well as qualitative features (e.g. concurrency modeling).

CT-RTE relies on MARTE language and provides the appropriate editors that support MARTE model specification. These are a Value Specification Language (VSL) editor and a Clock Constraint Specification Language (CCSL) editor.

VSL is used to specify the values of constraints, properties and stereotype attributes particularly related to non-functional aspects. Actually, this expression language can be used by profile users in tagged values, body of constraints, and in any UML element associated with value specifications. VSL supports the following requirements:

- How to specify parameters / variables, constants, and expressions in textual form.
- How to define relationships between various parameters or constant values with the support of arithmetic, logical, relational, and conditional expressions.
- How to define different time values and assertions in UML.
- How to specify composite values such as collection, interval, and tuple values.

CT-RTE provides an advanced VSL editor with completion that assists the user for the specification of real-time and embedded expressions in models.

CCSL is a declarative language used to specify constraints between clocks. The specified constraints may then be validated on models.

In short, MARTE concepts used in CT-RTE may be used to qualify UML structural and behavioral application models of real-time and embedded systems. Those qualified models include the complete semantics related to the characteristics of real-time systems and are therefore ready to support either a system analysis (e.g. a schedulability analysis, a performance analysis) or a system synthesis (e.g. a code generation).

Modeling requirements

IFP real-time supervisor relies on a mix of off-line and on-line scheduling and consequently deals with various time representations and real-time constraints. From a modeling point of view, it is important to represent the timing behaviors. In addition, the future versions of the supervisor will have to exploit the capabilities of multi-core architectures. Consequently, IFP is looking for modeling solutions which could provide a complete design framework to cope with these new challenges. The main need is to represent non-functional properties such as timing constraints. Then the possibility to use standardized patterns is also required in order to make the collaboration and exchange between different partners easier during the development. The utilization of a standardized design approach should also offer new opportunities in term of analysis tools. Finally, IFP pays attention to the high level of expressiveness of the descriptions achieved. Consequently CT-RTE has to offer comprehensive capabilities for the description of non-functional properties but at the same time it has to produce easy-to-understand models.

Assessment criteria

Following the IFP modeling requirements presented above, CT-RTE component was evaluated according to four criteria. Criterion #1 is relevant to the descriptive capabilities offered by the component. Can we just use standardized patterns of non-functional properties or is a strong language (with no limitation of modeling possibilities) available to specify non-functional properties ? Criterion #2 is the user-friendliness of the language ("easy-to-use" aspect). Is it a new language, a new methodology or philosophy or on the contrary is it intuitive ? Criterion #3 completes criterion #2 since it concerns the "easy-to-read" aspect. We want to specify more complete models, we do not want them to be more complex. The last criterion #4 is relevant to the availability of new tools that might be available for model analysis and transformation.

Results of assessment

During the modeling process performed by IFP, CT-RTE was used in different ways at different abstraction levels. Actually, CT-RTE component introduces some general concepts which are out of the scope of MARTE which is focused on the modeling of real-time systems. Such general concepts are for instance the following stereotypes: << tupleType >> which supports the definition of composed types, <<choiceType>> which supports the instantiation of an object which may belong to different classes and <<collectionType>> which supports the description of sets. CT-RTE component contains also more specific concepts. For example, it contains a MARTE library which defines various patterns allowing designers to represent periodic and aperiodic entities. Different time units are available too. At last, VSL language offers a real freedom of description for defining new specific values of non-functional properties.

Summarising, CT-RTE provides (i) general concepts which could belong to a more general standard, (ii) specific and ready-to-use concepts which improve the modelling of real-time systems and (iii) an open standard for value description which allows designers to specify their own non-functional properties. Criterion #1 which is relevant to the variety of description possibilities is therefore fully satisfied. The models using CT-RTE are clear and easy to understand, and consequently CT-RTE satisfies criterion #3. Nevertheless, even if the MARTE library is easy to use, the other stereotypes may require more documentation to be fully used. We expect a "how to" documentation to be issued in a not too far future and therefore criterion #2 will be met. Finally, we investigated the availability of external tools. Since MARTE adoption as an OMG standard is recent (June 2007), few MARTE-compliant tools are available yet. However the fact it is standardised will facilitate the arrival of tools. In the OpenDevFactory project for instance, Thales TRT showed how to generate a file compatible for RapidRMA[3] tool which supports the verification of schedulability properties. Evaluation criterion #4 should also be met in the future.

3.2 CT-PDM component

Description of component

CT-PDM stands for Technical Component for Platform Description Modeling.

CT-PDM covers software and hardware platform modeling. This section gives a brief presentation of the concepts and principles used for modeling these two kinds of platforms.

[3] http://www.tripac.com/

The MDA guide [3] provides the following generic definition of the platform concept: "*A platform is a set of subsystems and technologies that provide a coherent set of functionalities through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented*". Although this is a very broad and high-level definition which leaves a large scope for interpretation, it shows clearly that the MDA guide considers a *platform* as a support for the execution of software applications. This is consistent with the intuitive definition of "platform" used in the industry which refers to machines or systems such as frameworks, middleware, virtual machines and RTOS, which are built to support an execution process.

In [4], A. Sangiovanni - Vincentelli states that resources and services are provided by application programming interfaces (APIs). APIs should provide a complete and accurate description of the platform, so that the execution of any application consistent with these interfaces is guaranteed on the platform.
A model of a platform is thus a model of the platform APIs. Hence, a platform metamodel is nothing but a language to describe the APIs. A language dedicated to the modeling of Real-Time and Embedded (RTE) multitasking platforms was proposed : this is the UML Software Resource Modeling profile (SRM).
SRM has been implemented as an extension to the Unified Modeling Language (UML) (e.g. a UML profile). SRM profile is now a part of the new UML profile for MARTE. It is not a new set of multitasking APIs. It is rather a language to describe multi-tasking APIs (existing or under definition). SRM language has been built from the results of a detailed analysis of main multitasking API standards [5] and of several industrial standards. Three families of concepts were identified:

- Concurrent (i.e. parallel) execution contexts such as an interrupt context and a task context.
- Interactions between concurrent contexts for either communication or synchronization purposes (e.g. mailbox and semaphore mechanisms).
- Hardware and software resource brokering concepts, such as driver or memory management.

For this purpose, SRM profile is based on the "Resource Service" pattern. That pattern supports the description of resources having *properties* and providing *services*. Some properties and services might play *roles* which are modeled as resource attributes. Figure 3 illustrates how such a pattern is represented in SRM profile package. A schedulable resource in SRM is the modeling of a context for executing concurrent sequences of actions (i.e. a task). A schedulable resource has some attributes. Some of these attributes may play the role of the priority. A schedulable resource provides services also. Some services may be used by the resource itself to activate it. It would be too lengthy to describe all SRM profile details. Therefore the interested reader will find more information in [5].
The way SRM profile is used in the modeling process consists in three basic steps. First step is to model the multitasking APIs under consideration as a UML model library. Figure 3 below shows the first step of the (partial) modeling of the OSEK/VDX platform. Then in the second step, the designer applies SRM profile to clarify the taxonomy of modeling elements. For instance when modeling the OSEK/VDX platform, the OSEK/VDX BasicTask is considered as a SRM software schedulable resource whose attributes and operations are referenced in the stereotype properties. Finally in the third step, the classifier is instantiated for describing a multitask design (e.g. a task t1 is created in the application model). The resulting multitasking design is the platform instance model.

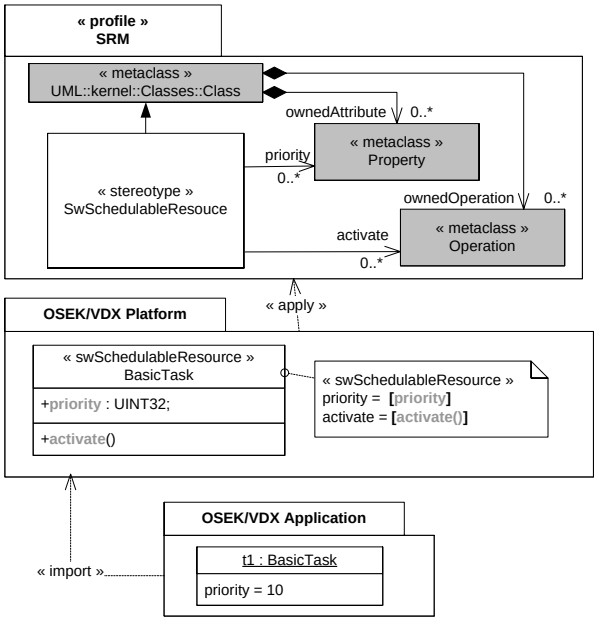

Figure 3: The "Resource Service" pattern

During the project, SRM profile was used by TRIALOG to model the RTAI operating system. The objective of the work was to create a UML model library that captures the RTAI concepts and that could be used by designers to describe how the functional part of various applications is implemented on a RTAI platform. We expect that in the future designers will have access to libraries allowing them to choose between different real-time operating systems such as Xenomai, VxWorks, RTX, RTAI, RT-Linux, etc. The development of the model libraries began by studying RTAI operating system APIs which are C-language oriented. However concepts like profiles, models or stereotypes are closer to Java. Therefore the first step of the model

development consisted in modeling RTAI APIs from an object-oriented perspective.

Each entity manipulated by the APIs has been defined as a class and the functions have been transformed into class methods. Then, the classes are able to apply the stereotypes defined in the profiles. It is also necessary to identify the functions of the stereotypes applied with the methods defined in the new class.

In addition, RTAI library methods require specific attributes which do not exist in SRM model. The new primitives can be created however inside the model library allowing the methods to use them.

For IFP, the goal was to use RTAI model library to describe the implementation of the engine test-bed functional specification. The approach used is a general design method based on three models: PIM, PDM and PSM. PIM, or Platform Independent Model, is the functional description of the application. No hypothesis about the final execution platform is performed at this step. PDM, or Platform Description Model, corresponds to the platform description. No functional aspects are represented. RTAI model library is an example of such a PDM. And last, but not least, PSM or Platform Specific Model describes the way the elements from the PIM are run by PDM components. **Figure 4** summarizes the approach.

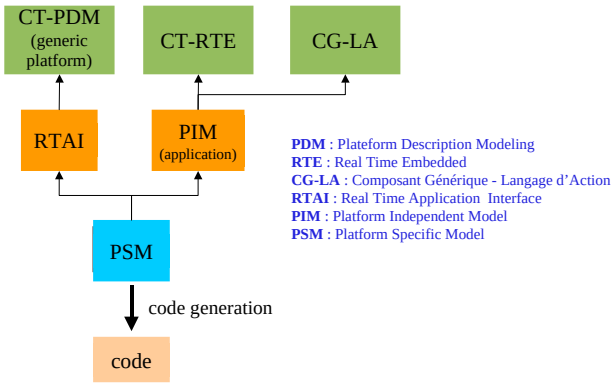

Figure 4: The utilisation of the modeling components by an application

We have then investigated how to use the automatic code generation from a model such as the PSM. TRIALOG has implemented generation scripts using the Acceleo tool [6]. The result of the generation is a set of classes that create, initialize and activate the tasks and timers defined in the PSM. The classes are written in C++ language and are specific to RTAI,

In addition to the description of software platforms, CT-PDM provides support for hardware platform modeling. For that purpose, CT-PDM relies on MARTE Hardware Resource Modeling profile (HRM). HRM is a model-based language for hardware platform design that can describe many low-level details.

HRM is composed of two views: a logical view that classifies the hardware resources depending on their functional properties, and a physical view that concentrates on their physical nature. Both are specializations of a more general model. The logical and physical views are complementary. They provide two different abstractions for the hardware and they could be simply merged. Each view is, in turn, composed of many models, as shown in Figure 5.

Stereotypes introduced within HRM profile are organized under a tree of successive inheritance paths from generic to more specific stereotypes ; no stereotype is an orphan. This is the reason why HRM profile is able to describe low-level details. Optional tagged values and composition mechanisms are strengthening this ability as well. Another feature of HRM profile is that it supports the description of most hardware concepts thanks to a wide range of stereotypes and to its layered architecture. If no specific stereotype corresponds to a particular hardware component, a generic stereotype may match instead. This is appropriate to support new hardware components and new technologies. Reader should refer to [1][6] for a more detailed description of HRM.

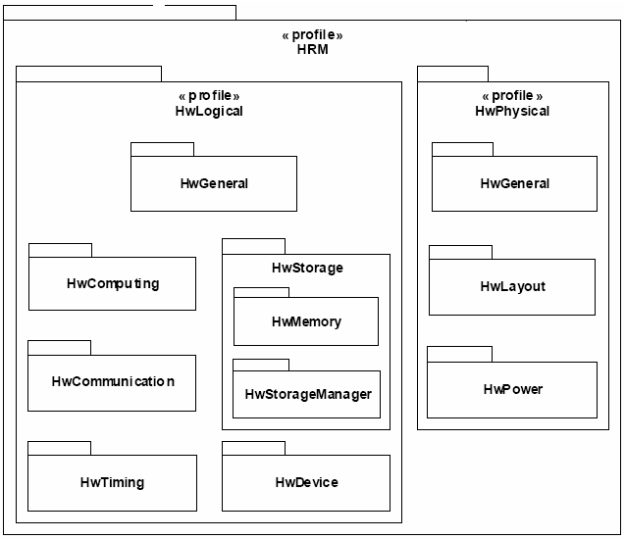

Figure 5: Hardware Resource Modeling models

Modeling requirements

IFP requirements for CT-PDM components were relevant to the modeling of the software and hardware platforms on one hand and the modeling of the implementation on the other hand. In addition to the assessment of CT-PDM modeling capabilities, IFP was also interested in CT-PDM capabilities relevant to the testing of the implementation and the support for multiple platform development.

Assessment criteria

IFP wanted to assess first whether HRM and SRM profiles provide an easy way to represent the engine test-bed specific architecture. An easy way means both the ease of utilisation and the availability of entities allowing designers to describe precisely the platform under consideration.

Then IFP wanted to use also CT-PDM to describe the implementation onto RTAI software platform and on the actual hardware platform. It was important for IFP that this part of the model be expressive and easy to understand.

Finally IFP wanted to assess how easy or difficult it was to go further into the modelling process towards code generation and multiple platform development.

Results of assessment

First, both SRM profile and HRM profile are very complete with respect to modeling entities. No lack was identified when modeling RTAI thanks to SRM stereotypes that map well every single concept of real-time operating systems. HRM profile proposes a large set of hardware concepts and could be employed in other domains than the industrial control system under consideration in the project. Future work at IFP will apply HRM profile to scientific computing for the description of repetitive structures (Repetitive Structure Modeling, annex E of the MARTE standard).

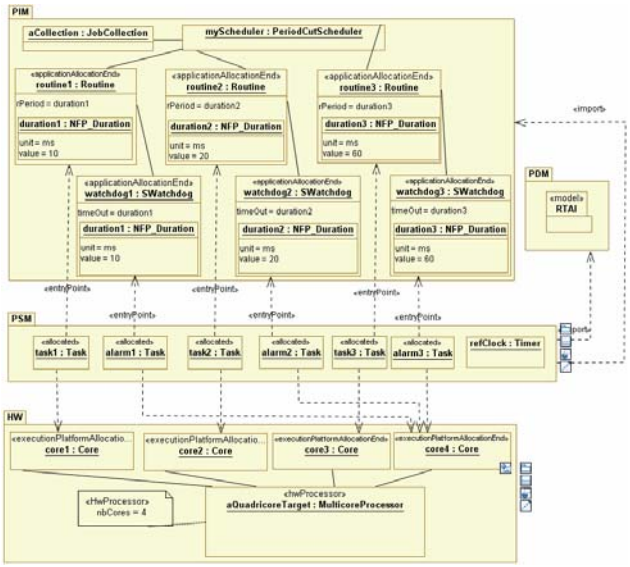

Figure 6: Example of implementation description with CT-PDM and CT-RTE

Concerning the implementation description, IFP considers it would be preferable to group all the implementation concepts into a unique profile. Actually IFP used a combination of SRM profile (CT-PDM) and Allocation profile (CT-RTE) and obtained interesting results by using the two concepts in a complementary way. For instance, SRM profile allows a designer to connect a task entity to an object of the functional description and to specify the called class operations. The Allocation profile allows a designer to stereotype the "grain" object, i.e. the entity he/she has to allocate in the functional description as well as in the software resources (task, timer, etc.) and to define the destination of the allocation (processor, memory, etc.). Figure 6 shows an example of such an implementation. Three levels of description can be distinguished, from top to bottom : functional description entities, software resources and hardware resources. Stereotyped dependencies allow designers to describe operation calls and resource allocation.

In brief, SRM and HRM profiles are efficient to describe software and hardware platforms. CT-PDM and CT-RTE components include interesting concepts for the implementation description, although a more comprehensive documentation would be appreciated.

At last, for the use case under consideration, TRIALOG performed some work on the development of a RTAI-oriented code generator which analyzes the implementation model and generates the corresponding allocation. Since MARTE is now a standard, the number of available real-time operating system description will certainly grow up. The creation of a central repository containing these descriptions would help. Consequently, an interesting work to be done in the future would consist in defining a template for the code generator. The template would avoid re-developing a code generator from scratch each time a real-time operating system is added to the repository.

IFP considers also that a potential advantage of CT-PDM is to make multiple platform development easier. Starting from a unique implementation model, the software code could generated automatically for RTAI, RTX and VxWorks operating systems. Some work is still needed in this direction however.

3.3 CG-LA component

Description of component

CG-LA stands for Generic Component for Action Language modeling. The component aims at providing a way to model 100% of the application behavior in UML. As shown on Figure 7, UML offers several packages for behavior modeling.

Use Cases specify requirements of the system, in other words they specify what the system must do. *State Machines* are generally used to specify behavioral control aspects of applications. *Interactions* are used to capture protocols of message exchange between instances composing the system. Activities are built using *Actions*. Actions are the basic entities for behavioral modeling in

UML. The UML specification provides a concrete syntax associated to the abstract syntax for all the behavioral packages except for the Action package for which UML specifies only the abstract syntax of Actions. Without a concrete syntax however, the Action package cannot be used for modeling purpose, Hence it is not possible to construct UML models that are completely specified in UML. Usually UML-based design approaches rely on programming languages such as C++ or Java to specify the behavioral algorithmic aspects [7]. This results in a mixing of the higher levels of abstraction and the semantics provided by UML with the lower levels of abstraction and the semantics provided by the programming languages. Such a mix may lead to inconsistencies in the resulting model. To cope with this problem, the Object Management Group (OMG) issued a RfP (Request for Proposal) that aims at standardizing a text-based concrete syntax for UML Actions abstract syntax [8].

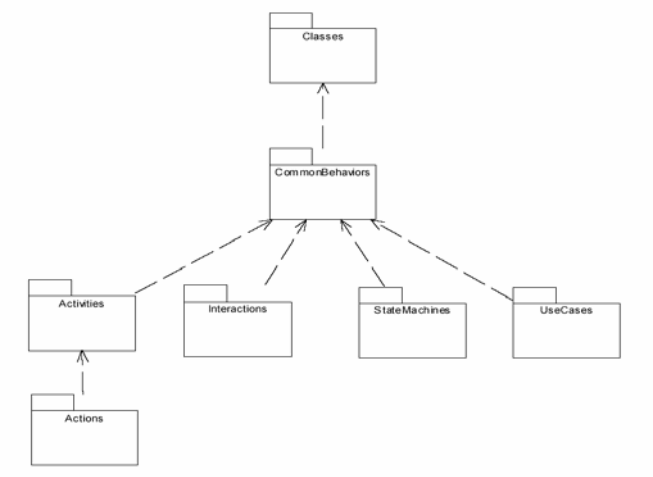

Figure 7: UML2 Packages for Behavior Modeling

CG-LA provides an action language named Accord-AL [9] that gives a concrete syntax for UML Actions abstract syntax. This language is defined by a one-to-one mapping between its elements and the elements of the UML Actions abstract syntax. Accord-AL language gives to UML the capability to use UML Actions for the algorithmic aspects of behavioral modeling. The concrete syntax allows designers to model 100% of their application behavior in UML, and thus avoids the mixing of different semantics and abstraction levels in the same model.

Since the concrete syntax is not standardized yet, no existing UML tool supports it for the time being. This is why CG-LA provides such a support for Accord-AL through an advanced editing tool. The editor supports the description of operation bodies using the Accord-AL language. The editor provides syntax highlighting and completion functionalities that helps the model designer describing behaviors with Accord-AL language. The specified behavior is then translated in UML2 actions. By doing so, we obtain a platform and language independent UML2 model that can be used for application execution or analysis purposes through simulation or code generation processes.

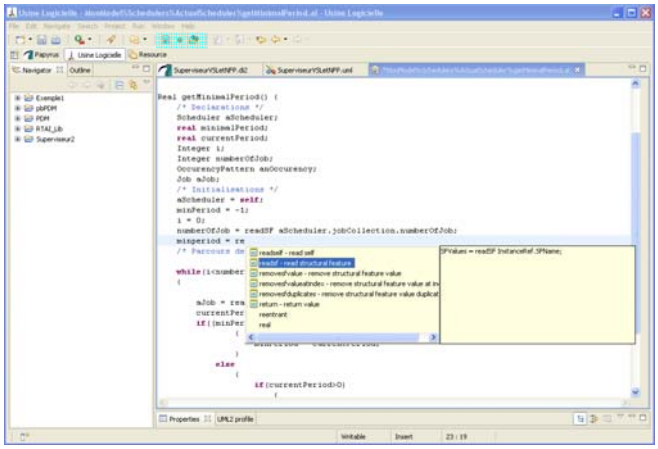

Figure 8: CG-LA- Action Language editor

Modeling requirements

IFP was interested to test an alternative approach to activity diagram and direct programming language description. IFP wanted to use an action language instead to describe the class operations during the modeling process.

Assessment criteria

IFP investigated the possibilities and limitations of the action language semantics and compared the utilization of the action language in place of activity diagrams. Since CG-LA is a textual editor, IFP paid as much attention to the user-friendliness (ergonomics) of the editor as to the descriptive power of the Action language.

Results of assessment

CG-LA was used to describe more than thirteen operations of the real-time supervisor model. IFP tried to use it each time C++ was not necessary. For several operations, IFP specify activity diagrams too.

The textual editor provided with the action language (see Figure 8) is considered pleasant. It offers syntax highlighting and textual completion on language keywords. Since it is not totally coupled with the Papyrus modeler yet, the completion does not include the names of instances and classes.

The Action Language semantics is object-oriented and close to UML2 concepts. Consequently, the utilization of the Action Language is very intuitive. Moreover, the component includes a documentation for each keyword with an equivalent example in

activity diagrams. Nevertheless, IFP assessment is that the semantics of the Action Language could be improved. For example, the semantics does not include any iterator. This is somewhat surprising because an action language is used in particular to explore models (classes, associations, etc.) and a keyword like "for each" could be helpful to apply the same action to every object of an association link.

The comparison with activity diagrams was very informative. Although they are well adapted to describe an object flow or a sequential action, activity diagrams become complex as soon as control structures (loop, if then else statements, etc.) are present. The approach followed usually by designers in such a case consists in dropping the modeling abstraction and using directly the programming language. An action language offers a real alternative close to the activity diagram philosophy but with the advantage of imperative textual language in term of control structure description.

In our case, the final implementation programming language is known and determined, but we need to underline that an Action Language improves the reusability of the model. Moreover, the use of CG-LA component and the associated Action Language is not limited to the modelling of real-time systems and they could be employed in every UML2 project.

4. Autocoding from UML models

As shown on the Figure 4 above, the engineering process used in the project included the code generation from the UML model. The code generation proceeds as follows. In addition to the scripts developed for generating the skeleton of the application (headers and source files), a script based on the Acceleo tool[4] generates a C++ RTAI file for the initialization of the application. This script reads the PSM model and for all real-time objects such as the tasks, timers and resources used by the application the script generates the code for the declaration, creation and deletion of the real-time objects. For example, for a task, the script uses information like the stack size, the task priority, the task entry point and whether the task is periodic or not for creating the corresponding task object within RTAI and for launching the task (or not) at initialization time.

5. Conclusion

Recently adopted, the OMG MARTE standard offers a new modeling solution for the modeling of real-time systems. In the OpenDevFactory project, IFP proposed a use case to assess the following three modeling components: CG-LA, CT-RTE and CG-PDM. The use case was relevant to the modeling of a real-time supervisor of engine test-bed which was complex enough to raise different modeling issues and consequently to explore different modeling solutions.

It appeared that the modeling components (profiles) dedicated to the modeling of software and hardware platforms (CT-PDM) and to the modeling of non functional properties (CT-RTE) seem exhaustive and offer an unlimited number of modeling solutions. These profiles are available with libraries of predefined entities that can be used as such for what could be named a standard modeling process. An interested observation is that HRM profile and CG-LA editor are not limited to the modeling of real-time systems. Finally, the Papyrus modeler, in the Eclipse-based integration platform was pleasant to use.

Some more work needs to be done in two directions. First, it is necessary to define a global modeling methodology that would describe the successive steps of the modeling process and include documentation and examples. The interaction between tools needs also to be improved in order to fully exploit the MARTE standard. Second, CT-PDM opens the way for multiple platform development and code generation. It is therefore now necessary to build and collect the descriptions of various operating systems based on SRM profile and to streamline the code generation process by providing some sort of a generic code generator using a CT-PDM model as input.

[4] Acceleo is a tool of the French OBEO company which translates an UML model to text.

6. References

[1] OMG: "*A UML Profile for MARTE*", ptc/07-08-04, 2007.

[2] OMG: "*Unified Modeling Language: Superstructure*", formal/2007-02-05, 2007.

[3] OMG: "*MDA Guide*" (version 1.0.1), Object Management Group, Inc., Needham, MA 02494, June. 2003. OMG document number: omg/2003-06-01.

[4] A. Sangiovanni-Vincentelli and G. Martin: "*Platform-based design and software design methodology for embedded systems*", Design & Test of Computers, IEEE Computer Society, p.23-33, 2001.

[5] F. Thomas, S. Gérard, J. Delatour and F. Terrier: "*Software Real-Time Resource Modeling*", Forum on Specification and Design Languages (FDL) 2007, ECSI, Barcelona, Spain, September 2007, p.231-236.

[6] www.obeo.fr

[7] S.Taha, A.Radermacher, S.Gerard and J-L. Dekeyzer: "*An Open Framework for Hardware Detailed Modeling*", In IEEE proceedings SIES'2007, pages 118-125, Lisboa, July 2007.

[8] B. P. Douglass: "*Real Time UML - Advances in the UML for Real Time Systems*" (3rd edition), Addison Wesley ed, 2004.

[9] OMG: "*Concrete Syntax for a UML Action Request For Proposal*", ad/2007-08-02, 2007.

[10] C. Mraidha, S. Gérard, Y. Tanguy, H. Dubois, and R. Schnekenburger: "*Action Language Notation for Accord/UML*", CEA, 2005.

7. Glossary

UML:	Unified Modeling Language
MARTE:	Modeling Analysis of Real-time and Embedded systems
CT-RTE:	Technical Component for Real-Time and Embedded applications modeling
CT-PDM:	Technical Component for Platform Description Modeling
CG-LA:	Generic Component for Action Language modeling