ISSN 1392-0561. INFORMACIJOS MOKSLAI. 2009 50

Conception of a Multi-Platform System Software and Firmware Development Tool

Mindaugas Vidmantas

Kaunas University of Technology,
Computer Engineering Department,
PhD student
Studentų g.50, LT-51368 Kaunas, Lithuania
E-mail: minvidm@ktu.lt

Egidijus Kazanavičius

Kaunas University of Technology,
Head of Computer Engineering Department,
Prof., PhD
Studentų g.50, LT-51368 Kaunas, Lithuania
Tel. (+370 37) 30 03 86
E-mail: ekaza@ifko.ktu.lt

This article proposes a new conception of the multi-platform software and firmware development tool. This conception is aimed at improving the quality as well as increasing the development productivity in multi-platform systems. The Model Driven Architecture (MDA) offers a more efficient software engineering process by raising the level of abstraction. A semi-formal Unified Modeling Language (UML) and its extension Systems Modeling Language (SysML) can bring software developers to a more formal model description, especially for real-time applications, enhancing possibilities to model the dynamics of the software model and decomposition of the physical architecture. The illustration of this tool conception is based on creation of a VoIP (Voice over IP) system.

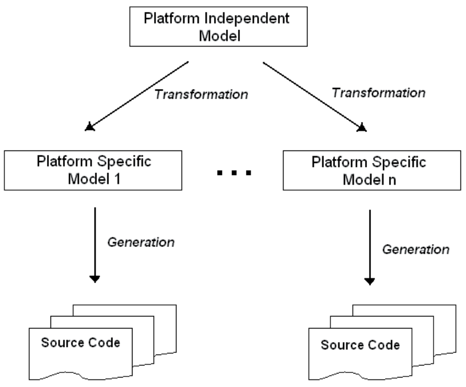
1. Introduction

Nowadays many visual modeling-based software development tools are designed for a very specific target to build single application, but not complex firmware. In the general case we propose firmware production software for the operating system (OS) X and hardware architecture (HA) Y, using the Model Driven Architecture (MDA)(OMG MDA) based on the Systems Modeling Language (SysML) and Unified Modeling Language (UML).The latest version of UML (2.2) (OMG UML, 2009) and SysML (1.1) (OMG SysML 2008) now enables us to more precisely model software for embedded devices (Hause, Thom, 2008). The SysML overcomes the limits of UML as a system modeling language (Colombo, Del Bianco, Lavazza, Coen-Porisini, 2007). The UML and SysML together with the technique of source code generation become a critical skill in a successful rapid software and firmware development. We need to admit that the general model of the proposed system consists of two models: a model of static structures and that of dynamic structures and behavior. Due to lack of space, this paper does not present the mapping of all SysML elements, just highlights some features from SysML that we found to be important, in this case, for VoIP system modeling, without going deeply into the usual UML and SysML style design and implementation details. In this paper we concentrate mostly on modeling of static structures, presenting the SysML ability to establish a relationship between the model and mathematical algorithm mapping. VoIP is one of many real-time systems, on which we have been working for the past few years. Its development process needs to be raised into a higher level of abstraction and to develop techniques in order to improve the quality as well as increase the development productivity. Details of the development tool conception implementation details are presented in Chapter 3.

1.1. MDA

The Model-driven development promotes the role of models, allowing developers to focus on the essential aspects of the system. MDA models are generally divided into three categories (Ortiz, Bordbar, Hernandez 2008):

Platform-Independent Models (PIM), Platform-Specific Models (PSM), and Code Layer. PSM takes the responsibility of describing the functionality and behavior in one or more particular technologies.



Platform Independent Model
Transformation
Transformation
Platform Specific Model 1
. . .
Platform Specific Model n
Generation
Generation
Source Code
Source Code

Figure 1. ***MDA hierarchy***

The proposed conception has a set of transformation rules in order to transform PIMs into PSMs and the latter into the final application code. Figure 1 shows MDA specification, which consists of a PIM-based SysML and a UML model, plus one or more PSMs and interface definition sets, each describing how the base model is implemented on a different platform. Finally, PSM is generated to a compatible language (Java, C++ or other) source code. For PIM modeling we used SysML and UML. In the VoIP PIM case, SysML is also used for a dynamic system model, performance simulation model, analytical model, and a system verification model. The PSM for the middle tier, which we call the Java model, is written in a language that is a version of UML. It uses classes, associations, and so on, as in UML, but there is a number of stereotypes, defined explicitly for the Java platform.

1.2. SysML

The OMG Systems Modeling Language (OMG SysML™) is a general purpose modeling language for system engineering applications. It establishes a description pattern for a great variety of complex systems. These systems may include hardware, software, data, methods, people, facilities, instruments and other elements within the physical environment. The SysML reuses a subset of UML 2 concepts and diagrams and extends them with some new diagrams and constructs appropriate for systems modeling (SysML). Compared to UML 2, the SysML has two new and three modified elements (Friedenthal, Moore, Steiner 2008).

1.3. MDA Generators

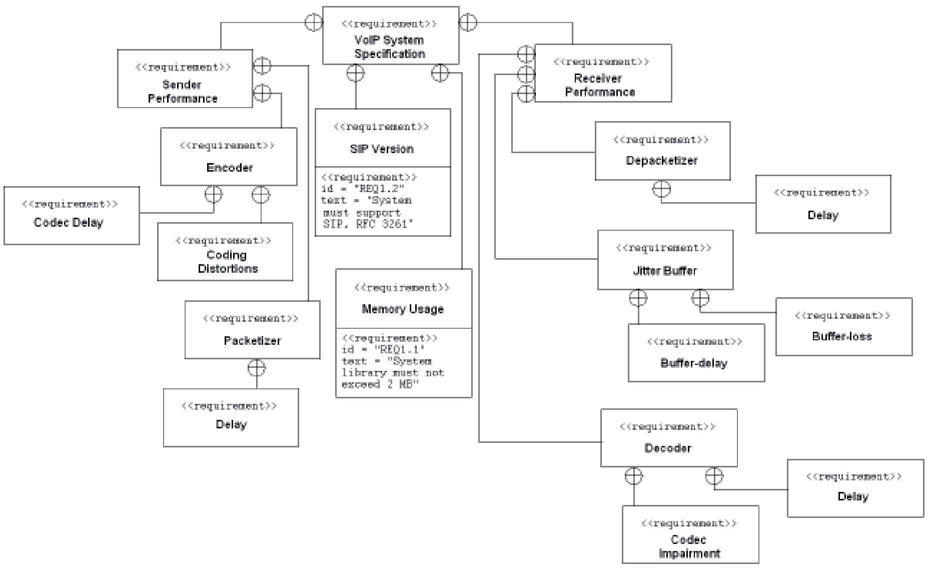
The purpose of MDA generators is to provide code generation facilities in order to keep the focus on the model itself and not on its implementation details. There are many projects based on MDA generation (CODE GENERATION). We chose the Eclipse Modeling Framework (Moore, Dean, Gerber, Wagenknecht, Vanderheyden 2004), which can be used to describe and build a model. Code generation advantages: Quality, Consistency, Productivity, and Abstraction.

2. VoIP System development using SysML

In this chapter, we demonstrate the VoIP system development using new SysML diagrams for a more formal model description, enhancing the ability to model the dynamics of a software model and decomposition of the physical environment. We describe models and their role in engineering the VoIP system.

2.1. Capturing the VoIP system Specification in a Requirement Diagram

Requirements are used to describe one or more properties or behaviors of a system that always have to be met and define the design problem being solved at various levels of detail. Figure 2 shows the requirements contained in



<<requirement>> VoIP System Specification
<<requirement>> Sender Performance
<<requirement>> Encoder
<<requirement>> Codec Delay
<<requirement>> Coding Distortions
<<requirement>> Packetizer
<<requirement>> Delay
<<requirement>> SIP Version
<<requirement>> id = "REQ1.2" text = "System must support SIP, RFC 3261"
<<requirement>> Memory Usage
<<requirement>> id = "REQ1.1" text = "System library must not exceed 2 MB"
<<requirement>> Receiver Performance
<<requirement>> Depacketizer
<<requirement>> Delay
<<requirement>> Jitter Buffer
<<requirement>> Buffer-loss
<<requirement>> Buffer-delay
<<requirement>> Decoder
<<requirement>> Delay
<<requirement>> Codec Impairment

F i g u r e 2. ***Requirement diagram showing the system requirements contained in the VoIP Specification***

the VoIP specification, which describes a contract between all those who create the system design and implement the system.

2.2. Defining the VoIP system and its External Environment Using a Block Definition Diagram

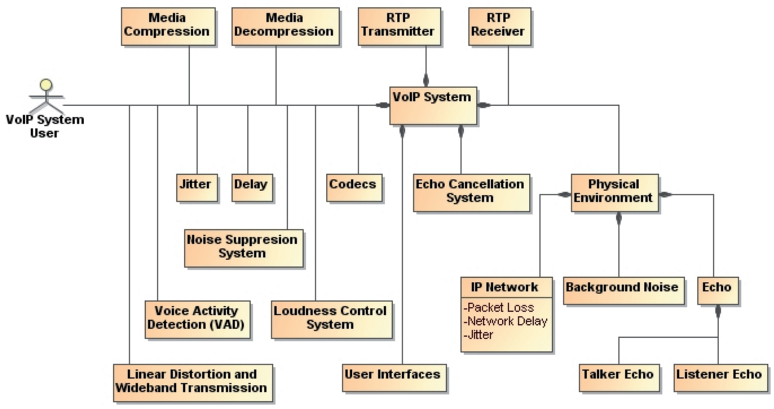
The block definition diagram is used to define blocks in terms of their features, and their structural relationships with other blocks and describes parts of the structure of a related system.

Figure 3 shows VoIP System Domain elements of the VoIP System, together with the physical environment and its users. VoIP system creators need to implement components according to the domain of the system.

2.3. VoIP system Context Using an Internal Block Diagram

The internal block diagram resembles a traditional system block diagram and shows the connections between parts of a block and looks at more detailed structure aspects than the block definition diagram.

Figure 4 shows the internal block diagram of the VoIP system fragment: connections between VoIP system data input, physical environment, echo cancellation system in more detail.



F i g u r e 3. ***Block definition diagram of the VoIP System Domain showing the VoIP System together with the physical environment and its users***

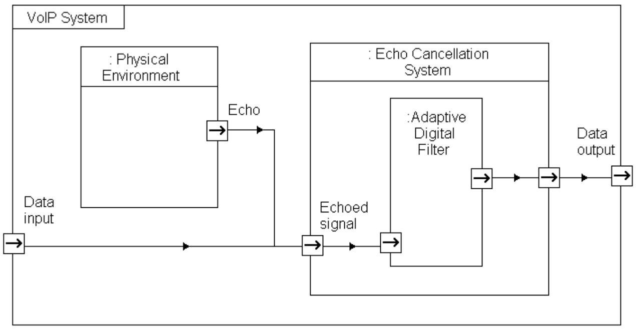


Figure 4. Internal block diagram of the VoIP system fragment

that overlay the functional decomposition in the PSM (MDA) model to capture the dynamics envisioned within the system. This tool consists of four main components: Visual Modeling System, MDA Generator, Parameter Observer, and Management System. First of all, a visual PIM has to be built using the Visual Modeling System. When the platform for deployment is specified, MDA Generator transforms PIM to PSM, and then according to PSM automatically generates a source code for appropriate OS. Figure 6 shows the overall structure of the proposed system.

2.4. VoIP System Parametric Diagram

The parametric diagram describes the relations between properties of different blocks. We can model such relationships to integrate the VoIP adaptive digital filter performance model in the VoIP system model.

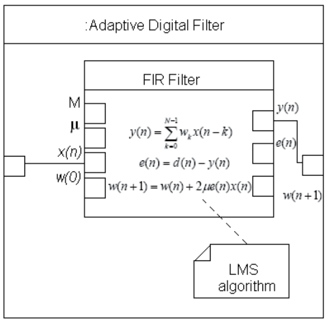


Figure 5. Parametric diagram of an adaptive digital filter

Figure 5 shows the parametric diagram of an adaptive digital filter and systems of equations that constrain the properties of blocks of the echo cancellation system. The constraints are expressed as equations whose parameters are bound to the properties of a system. The parametric diagram was created for modeling performance (Buede 2009).

3. System implementation

The proposed conception of the software development tool has a general set of graphical modeling, control, and cross-compile structures

After the whole necessary source code has been obtained, we can go over to the next stage – automatic firmware creation. Figure 7 shows the general firmware creation structure.

As a firmware development platform base we chose the Eclipse framework (ECLIPSE), which is a highly integrated and extensible tool platform. The proposed Eclipse plugin easily integrates together with EMF as a modeling framework, Graphical Editing Framework (GEF) as a graphical representation framework, and Ant (Loughran, Hatcher, 2007) as a build tool. EMF enhances the Meta Object Facility 2.0 (MOF (of the OMG)) Ecore model and restructures its design in a way that is easy for the developer. We extended EMF for PIM modeling with SysML and UML. The VoIP project consists of many modules. Module buildfiles are written in XML. Ant is used together with Ivy as a dependence manager. When the dependencies of a module are resolved, it means that Ivy has determined a complete set of dependencies for all configurations of the module. It has managed to locate all the artifacts, locally or remotely, and any associated metadata. Monitoring and logging of the build process is performed using Log4J (Log4J). When a firmware is built, primary testing is executed by the JUnit testing framework. After firmware testing, the monitoring component, residing in the Management System, after a few minutes delay, starts checking the state of the firmware by executing scenarios and formal proofs of correctness.

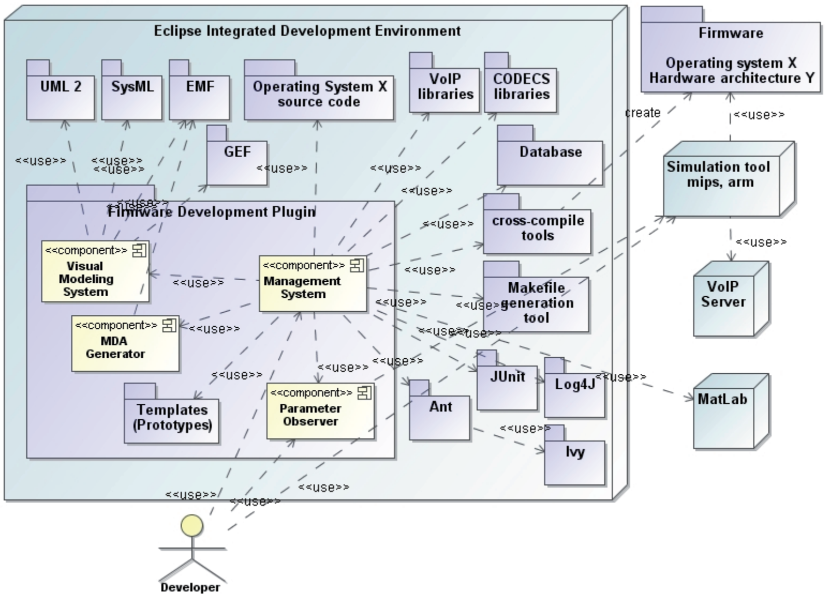

Eclipse Integrated Development Environment
UML 2
SysML
EMF
Operating System X source code
VoIP libraries
CODECS libraries
GEF
Database
Firmware Development Plugin
cross-compile tools
<<component>> Visual Modeling System
<<component>> Management System
Makefile generation tool
<<component>> MDA Generator
JUnit
Log4J
Templates (Prototypes)
<<component>> Parameter Observer
Ant
Ivy
<<use>>
Developer
Firmware
Operating system X Hardware architecture Y
create
Simulation tool mips, arm
VoIP Server
MatLab

Figure 6. ***Eclipse plugin realization scheme***

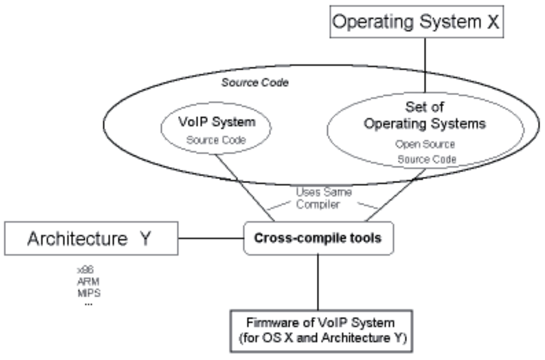


Figure 7. ***Firmware creation for OS X and HA Y with the VoIP integrated structure***

Extension points of required libraries are used to integrate them into the proposed plugin in a proper way for us. It is possible to access the plug-in extension registry using the *Plaform.getExtensionRegistry()* method. It contains plug-in descriptors, each representing a plug-in. The registry provides the following methods for extracting information about the various plug-ins without loading them.

The VoIP project needs external libraries (VoIP core, Voip communication and data transmission, third-party CODECs). The parametric diagram and related modeling information can be provided for appropriate simulation and/or analysis tools to support execution. Two integration methods are fairly common across all IDEs: external tool invocation and filtering (Herrington, 2003). The external tools in our proposal are: hardware architecture simulation tools, VoIP Server ant MatLab. Integration methods are used for data transmission, test scenario loading and for visualizing the collected data.

4. Conclusion and Future Work

This proposal presents a software development tool conception, which has a high level of abstraction of the software development (using UML together with SysML) visual modeling tool and can model the PIM. The rest, very complicated technically, work is done automatically: PIM transformed into PSM, after generating the source code, and then integrated and compiled with cross-compile tools for OS X and HA Y. Later the final firmware has been tested. After the test it is loaded on the HA Y simulator. In the case of VoIP, test scenarios with the VoIP server are executed to get data in XML format and to analyze and visualize the performance of the built system in MatLab by the external tool invocation method. On demand, the system parametric models can capture the constraints on the properties of the system that can then be evaluated by the same analysis tool.

Future work is to completely implement the proposed conception of the software development tool and extend this tool with artificial intelligence methods.

REFERENCES

FRIEDENTHAL Sanford; MOORE Alan; STEINER Rick. (2008). *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann, 2008. ISBN 0123743796.

HAUSE M.C.; THOM F.. (2008). ARTiSAN Software Tools. In *13th IEEE International Conference on Engineering of Complex Computer Systems*, 2008.

OMG MDA (Model Driven Architecture). (2003). MDA Guide Version 1.0.1. Available from http://www.omg.org/mda. [Accessed Jun 2008].

OMG SysML (2008). OMG Systems Modeling Language (OMG SysML) version 1.1. Available from www.omg.org. [Accessed Jan 2009].

OMG UML (2009). OMG Unified Modeling Language (OMG UML), Infrastructure version 2.2. Available from www.omg.org. [Accessed Mar 2009].

ORTIZ, Guadalupe; BORDBAR, Behzad; HERNANDEZ, Juan. (2008). Evaluating the Use of AOP and MDA in Web Service Development. *ICIW*, p. 78–83.

BUEDE, M. Dennis (2009). *The Engineering Design of Systems: Models and Methods*. Wiley 2009. ISBN 0470164026.

MOORE, Bill; DEAN, David; GERBER, Anna; WAGENKNECHT Gunnar; VANDERHEYDEN, Philippe (2004). *Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework*. IBM.

COLOMBO, Pietro; DEL BIANCO, Vieri; LAVAZZA, Luigi; COEN-PORISINI, Alberto (2007). A Methodological Framework for SysML: a Problem Frames-based Approach. *14th Asia-Pacific Software Engineering Conference*.

LOUGHRAN, Steve; HATCHER, Erik (2007). *Ant in Action*. Manning. ISBN 1-932394-80-X

HERRINGTON, Jack (2003). *Code Generation in Action*. Manning. ISBN 1-930110-97-9

ECLIPSE. http://www.eclipse.org.

CODE GENERATION. http://www.codegeneration.net.

Log4J. http://logging.apache.org/log4j/1.2/index.html

DAUGIAPLATFORMĖS PROGRAMINĖS IR SISTEMINĖS ĮRANGOS KŪRIMO PRIEMONĖS KONCEPCIJA

Mindaugas Vidmantas, Egidijus Kazanavičius

S a n t r a u k a

Straipsnyje pateikta daugiaplatformės programinės įrangos kūrimo ir integravimo į įterptinės sistemos sisteminę įrangą (ang. *firmware*), kūrimo priemonės koncepcija, kurios veikimo pagrindą sudaro generuojančios sistemos. Modeliavimas atliekamas pagal MDA architektūrą, naudojant UML ir jos pagrindu sukurtą SysML modeliavimo kalbą, kuri leidžia daug formaliau aprašyti modeliuojamą sistemą ir jos dinamiką. Programinės įrangos kūrimas iliustruotas balso perdavimo IP tinklais programos pavyzdžiu.