A New Models Transformation Approach for Embedded Systems

Mohamed AIT ALI
Lab. SIR - Mohammadia School of Engineers
Rabat, Morocco

Noureddine ZHAR
Lab. SIR - Mohammadia School of Engineers
Rabat, Morocco

Mohsine ELEULDJ
National School of Applied Sciences
Oujda, Morocco

ABSTRACT
Embedded systems are becoming more complex by integrating multiple features. They require a lot of resources to improve execution performances. Their developments are a real challenge due to both their complexity and their quality of service requirements. To manage this complexity, a model driven approach focuses on the design of these systems by raising the level of the specification abstraction. For this reason the number of modeling languages (metamodels) is increasing (scientific publications, industrial projects). However, there is currently little use and dissemination of good practice to define metamodels (metamodeling) and transforming these metamodels for verification, validation and code generation. However, the identification of a string of well-structured model transformation and formalization of metamodeling patterns should be an important practice in the sense that it should speed up the metamodels writing, facilitate their reuse, teaching and finally processing for code generation. The research below suggests a structure of a model transformations chain by defining an intermediate language.

General Terms
Embedded systems; metamodel; patterns; code generation.

Keywords
Metamodel; Model; Models' transformation; Intermediate language;

1. INTRODUCTION
To answer the increasing complexity of the embedded systems real-time, several languages of modeling such as UML (Unified Modeling Language), AADL (Architecture Analysis and Design Language) and SDL (Specification and Description Language) [1] have been developed in order to allow a global and abstracted approach of the complex computer systems modeling which require a modeling for the analysis, the check [2] and the validation of the systems [3] before any production of code for platforms which are generally very forced. Actually, the models' engineering (MDEiv) present a development paradigm based on the models use [4], however if the modeling languages allow to describe the deployment of application program bricks by considering an abstract support (medium) of execution, a very important phase of the development consists in concretizing these bricks by the executable code generation intended for a defined execution support (medium) [5]. For this purpose the engineering managed by the models (MDE) proposes a model transformation technique [6] to be able to generate (automatically or semi-automatically) application soft wares dedicated to particular technological platforms (computing and electronic), from a description independent from any technologies. Nevertheless, it is worth mentioning that there are several modeling languages [7] the number of which increases constantly and the consideration of the diversity of the jobs in the conception of the embedded systems goes through the integration of the various tools and the modeling languages used by these jobs. Every modeling language is represented by a meta-model which is only a model that describes the modeling elements. The target languages are also represented by meta-model. So, for the executable code generation, we transform models corresponding to meta-model towards meta-model target [6]. This requires high-level of expertise for the definition of several transformations and rules of transformation due to the significant number of the modeling languages, target languages and the big difference between the high level of abstraction of the modeling languages (UML, AADL, SDL) and the low level of abstraction of the target languages (Java, C, C ++).To resolve this problem, our research is interested in the definition of pivot modeling language or intermediate which we named COCODEL (Communicating Component Description Language). This language will have the particularity to define a models transformation chains structure dedicated to the development of the embedded systems in order to allow the executable code generation and the generation of the code for the formal check of the embedded systems.

In this purpose, this work is articulated as follow: First of all, we will present the general framework of the study to place our research work and to present its general aim, then we will enumerate the advantages of an intermediate language in transformation chain models, and finish with the definition and the organization of our intermediate language.

2. GENERAL FRAMEWORK
Our approach joins the MDE framework and recommends that instead of making direct modeling languages transformations (UML, AADL, SDL) towards the target languages of low level abstraction (Java, C, C ++), subject of several works [8] [9] [10] and several tools allow its achievement (ex: Rhapsody IBM), we are going to perform a first transformation towards intermediate meta-model. This one is composed of various common notions of the modeling languages (UML, SDL and AADL). The second transformation is to convert the result of the first one towards meta-model closer to languages of low-level abstraction (Java, C, C ++).

Thus, the objective is to have a chain of modular, reusable and evolutionary transformation to integrate new languages in entry and exit while obtaining a more optimized code into the embedded systems. For that we propose the following chains:

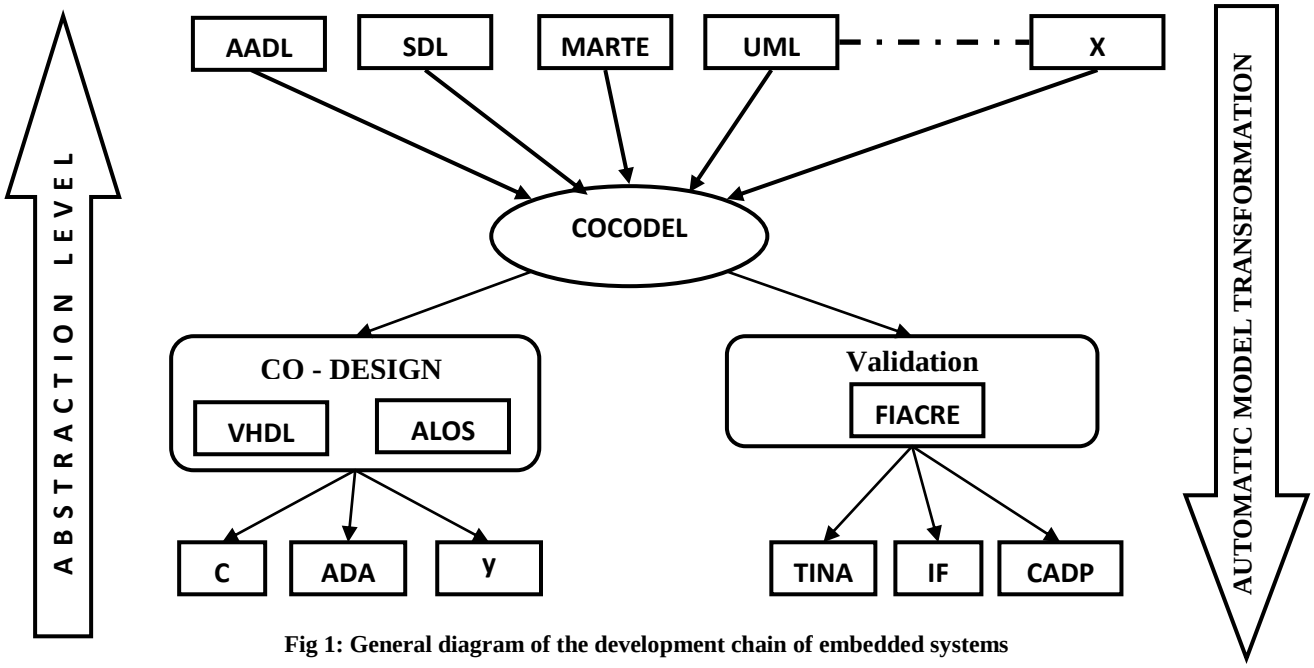

Fig 1: General diagram of the development chain of embedded systems

Fig 1: If necessary, the images can be extended both columns

3. ADVANTAGES OF INTERMEDIATE LANGUAGE

After a quite detailed study of the modeling languages we noted that they represent important meta-models with a significant number of classes and concepts [7] with many several similarities between the various languages. This noticed identical parts for the majority of the languages during the transformation of these meta-models. It was also noticed that the good semantic separation between architecture and the behavior in these languages. For these reasons, we introduced an intermediate language into the chain of transformation which we named Cocodel (Communicating Component Description Language) and of which we defined the meta-model. The introduction of an intermediate language has several advantages.

Indeed, the intermediate languages allow a generic description of the services and concepts used by the majority of the languages. The taking into account of a new language or a new executive is clearly simplified (figure 2) and requires a low number of concepts to be considered during the transformation. The capitalization of the common rules is direct. Moreover, we can carry out transformations of refinement and optimization directly on the level of descriptions of the intermediate language according to the concepts handled by each meta-model.

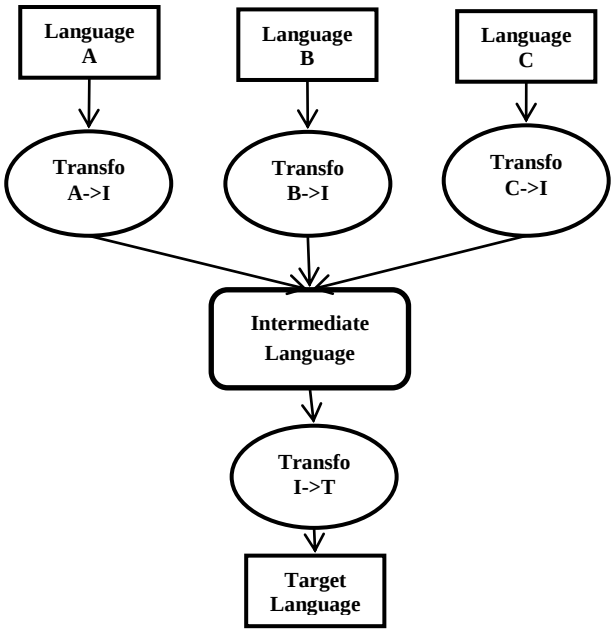

Fig2:Approach to model transformation with intermediate languages

4. COCODEL LANGUAGE DEFINITION

The purpose of this pivot language named Cocodel is to allow a generic description of the services and concepts used by the majority of the modeling languages which are very rich and too much used as the UML2, AADL and SDL languages.

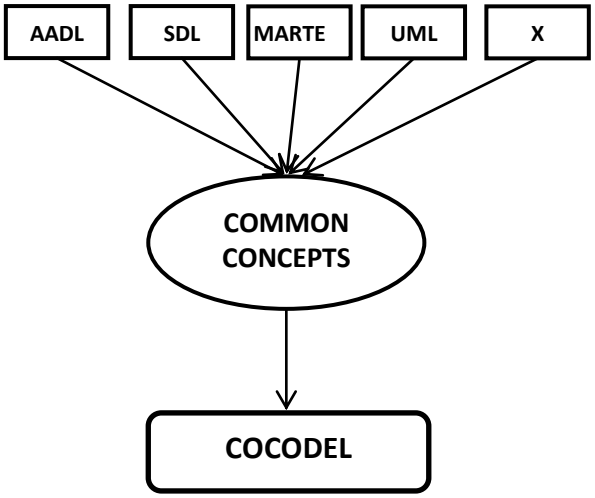

Fig3: Language Cocodel Description

During construction of the meta-model Cocodel, we had a choice between two strategies to define it:

• Identify common concepts between different modeling languages in order to have a minimal description of Cocodel.

• Assemble all possible concepts of modeling languages.

After studying the two strategies, we realized that given the large number of concepts offered by modeling languages, the description of Cocodel must be minimal. We must therefore incorporate the common and important concepts in these different languages for having in the end a generic description for the meta-model "Cocodel".

During the writing of the meta-model Cocodel, we used generic patterns that are often used in defining meta-models for existing languages, in addition to other reasons specific to particular areas. Reusing patterns saves time and quality.

A model of a system is considered as a set of components whose ports are connected by relations. The meta-model Cocodel is divided into four packages:

4.1. The package architecture

It describes the appearance of the architecture of the meta-model Cocodel. It defines the architectural abstractions based on the technology components. Indeed the system is described by a number of components. Its configuration is represented by an assembly of components via connectors through associated ports.

4.2. The package behavior

It is described by a state machine transitions through the class-state machine that describes the internal behavior of an object using a finite state automaton.

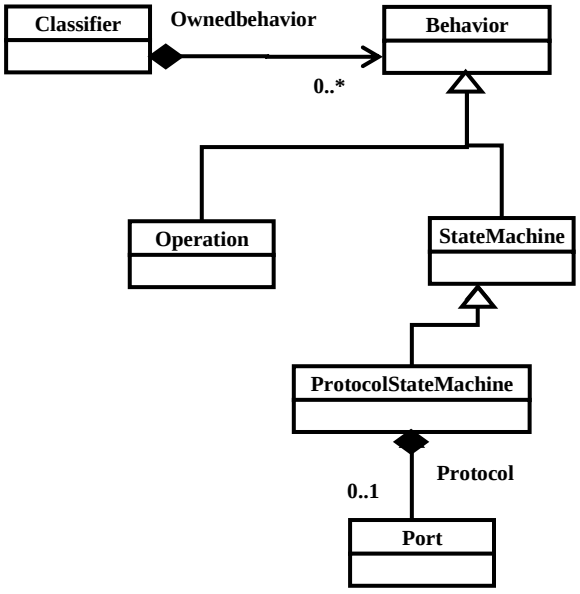

Fig4:Partial behavior Package (State Machine) of Cocodel

4.3. The package action

It defined a language-specific action Cocodel order to describe the body of all actions, operations, and expressions used in a model Cocodel, which take all the basic actions required to model embedded systems and are defined in [11] [12].

4.4. The package instance

It defines the representation of an instance of the architecture.

While allowing a particular representation system, classes of this package refer to classes in the package architecture. It is therefore dependent packages presented above.

The representation by instance allows, in particular:

• To have models whose root is an implementation of the system,

• To consider only object instances of Cocodel,

• To treat only property values relevant to code generation or specific analyzes.

5. THE MODELS TRANSFORMATION

Automating the transformation from one model to another is done by model transformation. This is done by matching elements of the entry model with elements of the model output. Consequently, a transformation is based on the meta-model source and target met-models handled. As well, transformation rules from one model to another will be specified to create the chain of transformations using the specific language QVT (Query, View and Transformation).

6. CASE STUDY

To validate our chain of embedded systems development via an intermediate language, we chose to perform a case study as a demonstration. The case study concerns a shooting simulator modeled with UML. The modeled shooting simulator consists of two main parts implemented in the software and the FPGA with the hard real-time constraints of

the tracking algorithm. We will apply a set of automatic transformations described in our chain, to validate the simulator and automatically generate executable code.

7. CONCLUSION AND PERSPECTIVES

During our research we have demonstrated that using the approach of model transformations in a structured and thoughtful, manages to build a chain of model transformation and reusable code generation that supports multiple languages by the introducing of the intermediate language (Cocodel). The Engineers' work is simplified by allowing them to focus on profession aspects rather than simple translations. It guarantees a level of quality and promotes reuse.

This study also tends to present the feasibility of automatic generation of executable code from a graphic description "UML2.0, AADL" to low levels of language because that most code generation tools and simulation from models do not always allow to do. If the approach simplifies the problem by manipulating only a few concepts, a possible improvement is to expand the number of concepts used to encompass all of the modeling languages in the chain. For this purpose, there shall be specified a real-time model [13] which covers the different notions and concepts of time in different modeling languages for real-time embedded systems, and define them in the Cocodel meta-model for modeling the time constraints related to these systems.

Taken into consideration that the code generated is generally not optimized, a perspective of this work is to think of a method for optimizing the generated code.

8. REFERENCES

[1] A. A. J.P. Babau, «A development method for Prototyping embedded SystEms by using UML and SDL (PROSEUS,» workshop SIVOEES , Hungary, 2001.

[2] J. R. F. B. P. Dhaussy, «Mise en œuvre d'unités de preuve pour la vérification formelle de modèles ,» IDM'07, France, 2007.

[3] H. Kopetz, Real-Times systems : Design Principles for Distributed Embedded Applications, USA: Kluwer Academic Publishers, 1997.

[4] OMG, MDA Guide Version 1.0.1, 2003.

[5] M. CHOUKRI, «Modélisation des systémes temps-réel embarqués en utilisant AADL pour la génération automatique d'applications formellement vérifiées,» France, 2010.

[6] OMG, «Mofqvt Technical report,» Object Management Group, 2005.

[7] OMG, Unified Modeling Langage :Superstrustur Version 2.0, OMG, 2003.

[8] O. Habart, «UML vers IF,» ENSIETA, France, 2004.

[9] U. Faghihi, «UML vers IF,» ENSIETA, France, 2005.

[10] T. Abdoul, «AADL vers IF,» ENSIETA, France, 2006.

[11] OMG, Action Semantics for the UML, Request For Proposal, OMG Document, 2003.

[12] OMG, UML Action Semantics, November 2001.

[13] A. KOUDRI, «Méthodologie UML/MARTE Pour La conception Conjointe Logicielle / Materiélle,» ENSIETA, France, 2010.