Benefits of the MDE approach for the development of embedded and robotic systems

Application to Aibo

Xavier Blanc[1], Jérôme Delatour[2], Tewfik Ziadi[1]

[1]Laboratoire d'Informatique de Paris 6 (LIP6)
104 avenue du président Kennedy
75016 Paris
Xavier.Blanc@lip6.fr, tewfik.ziadi@lip6.fr

[2]Equipe TRAME, CER ESEO
4 rue Merlet de la Boulaye - BP 30926
49009 Angers cedex 01
jerome.delatour@eseo.fr

ABSTRACT. *Model Driven Engineering (MDE) raises the level of abstraction of the development life cycle by shifting its emphasis from code to models and model transformations. According to the well-known principle of separation of concerns, MDE advocates the isolation of business concerns from their technical achievement. The idea is that the business concerns can be modeled independently from any platform concerns. Therefore, business models are not corrupted by technical concerns. In this way, the main part of the development becomes an activity upstream, dedicated to business concerns through the elaboration of the application model that abstracts away technical details, i.e., the so-called Platform-Independent Model (PIM). The transformation of a PIM into a Platform-Specific Model (PSM) is then achieved when introducing into the PIM the technical considerations depending on the chosen platform. We applied MDE to develop software systems for Aibo which is one of several types of robotic pets designed and manufactured by Sony. The objectives was (1) to analyze advantages provided by models in this specific robotic context, (2) to measure the maturity of MDE provided technologies and (3) to highlight the limitations of the approach . We present in this paper the approach we follow for applying MDE for Aibo and we present the results we obtained.*

KEYWORDS: *Model Driven Engineering, Robotic, Ingénieurie Dirigée par les Modèles, Aibo, Domain Specific Language, Langage dédié*

1. Introduction

Model Driven Engineering (MDE) raises the level of abstraction of the development life cycle by shifting its emphasis from code to models and model transformations. According to the well-known principle of separation of concerns, the MDE advocates the isolation of business concerns from their technical achievement. The idea is that the business concerns can be modeled independently from any platform concerns. Therefore, business models are not corrupted by technical concerns. This is strongly anticipated as a way of simplifying the construction of systems. In this way, the main part of the development becomes an activity upstream, dedicated to business concerns through the elaboration of the system model that abstracts away technical details, i.e., the so-called Platform-Independent Model (PIM). The transformation of a PIM into a Platform-Specific Model (PSM) is then achieved when introducing into the PIM the technical considerations depending on the chosen platform.

The MDE approach is quite often applied for large scale industrial systems. Indeed, MDE standards such as UML (Unified Modeling Language) and on-the-shelf transformations such as "UML to EJB" have been defined to specify and to build those large systems. However, the MDE approach has been defined to specify and to build any systems, whatever their size and their business domain. Standards such as the MOF (Meta Object Facility) and technologies such as model transformations and model validation can be used to define domain specific modeling language that can be used to build any.

In order to check if MDE can really be used to build systems other than large scale industrial systems, we tried to apply it to build software systems for Aibo which is one of several types of robotic pets designed and manufactured by Sony. Aibo is a reactive system that can move, see, hear and speak. Aibo software can be developed in order to make Aibo react (move and speak) when it receives events (hear and see). Building software for Aibo is a complex task as (1) Aibo is a complex platform with a lot of joins and sensors and (2) no industrial development environment is provided by Sony. Moreover, Aibo should be considered more as a family of robotics pets rather than a single robot. Each member of the family has its specificities. Aibo software should be then reusable as much as possible for all members of the Aibo family. For all these reasons, we considered that Aibo was an ideal platform to check if MDE principles can be applied to build all kinds of systems.

The next section of this paper presents MDE concepts used to define domain specific modeling languages. Based on those concepts, section three of this paper presents how we applied MDE for Aibo. Then our conclusion is presented in section four.

2. MDE Approach

This section presents the MDE principles and core concepts. Despite its relative youth, the MDE approach defines a long list of acronyms (MDA, PIM, PSM, CIM, MOF, EMF, QVT...) and slightly adapts the meaning of some existing concepts (model, transformation, meta-model...) to its needs. Moreover, different implementations of the MDE approach exist: « Software factories » from Microsoft [Greenfield 04], MDA (Model Driven Architecture) by OMG [Soley 02], EMF(Eclipse Modelling Framework) which is the MDA approach of Eclipse [Budinsky 03], MIC (Model Integrated Computing) from the DARPA [Frake 97] ... Each of them defining their own vocabulary and techniques, it is still a challenge to reach a general consensus on the MDE core concepts.

Therefore, we present the definitions that reach a large agreement in the MDA community. These definitions are based on the works made by the french group « AS MDA » [AS MDA 06].

2.1 Models, meta-models and meta-metamodels

The notion of model is quite old and has been defined before MDE. "A model represents reality for a given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality." [Rothenberg89]. It is a set of statements about something which is under study (the system). For instance, a map of the world is a model of the world. Depending on the authors, this relationship between a system and a model is called either, « RepresentationOf », « RepresentedBy », « Describes »... As stated by J.M. Favre, it is very important to understand that this relation is actually defined between systems: being a model or a system under study is a relative notion, not an intrinsic property of an artefact. In fact these notions are roles that a system can play with respect to another system [Favre 04]. It is obvious to consider the map of a world as a system and make a model of it (for instance, a sketchy drawing of this map).

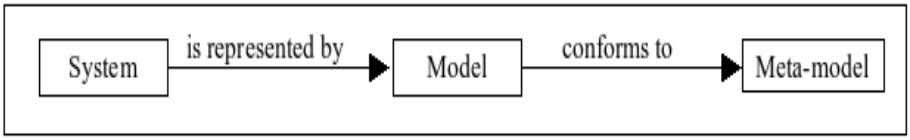

Figure 2.1: Relationship between system, models and meta-models

In MDE, another relationship is required for the models: they must conform to a meta-model. A meta-model is a precise specification of the considered concerns the model will represent. A meta-model of the map of the world could be the legend of

the map. In other words, a meta-model could be seen as a way to define the sets of possible models (a model of a set of models). A rough analogy with the language theory could be that a meta-model is one possible representation of the grammar of a language.

In each trend of the MDE, there is an attempt to define a meta-metamodel: a common language for defining meta-models. In MDA, they define a four-level modeling stack (sometimes called MDA pyramid) where:

- M0 is the real world,
- M1 is the model level,
- M2 is the meta-models levels, UML is a meta-model. In the next section of this article, a meta-model of the AIBO language will be presented.
- M3 is the meta-metamodel, it is self-defined and called MOF (Meta Object Facility).

A rough analogy with the language theory has been established in figure 2.2. The Extended Backus Norm Form (EBNF) could be seen as a meta-metamodel as it can define the grammar of different languages.

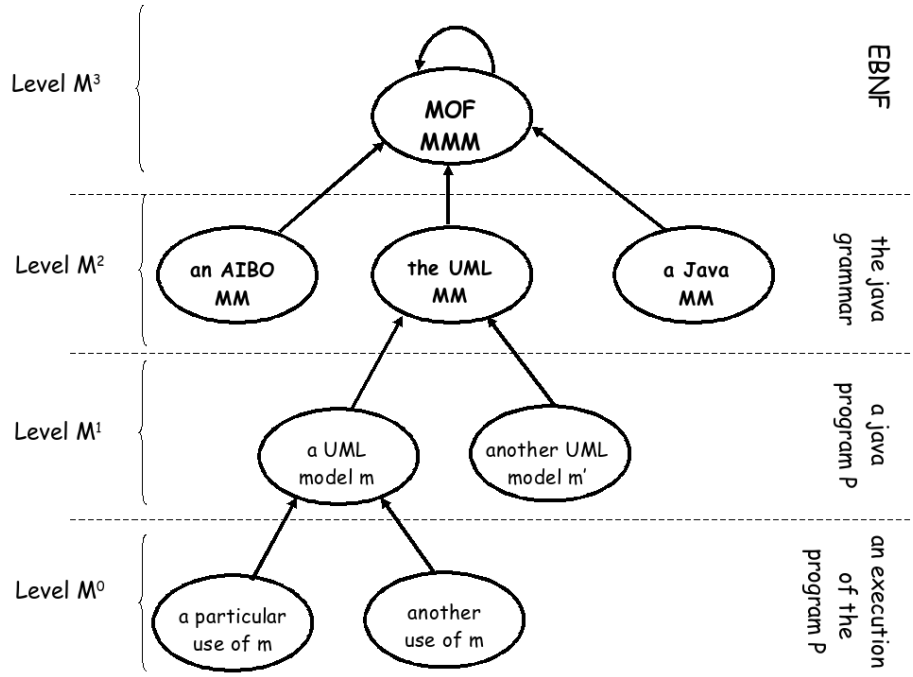

Figure 2.2 : MDA pyramid, adapted from [Bezivin 04]

MDA and other MDE trends facilitate the definition of DSL (Domain Specific language [Czarnecki 00]). It consists basically in defining a consistent meta-model and, thanks to the normalized MDA infrastructure, a model editor (textual and, in certain case, graphical) could be generated. The models are stored in an XMI format (an XML variant) and could be exchanged between MDE tools.

2.2 The notion of Platform

The MDA guide [MDA03] provides a generic definition of the platform concept "A platform is a set of subsystems and technologies that provide a coherent set of functionality 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". This is a very large high-level definition that leaves a wide scope for interpretation. There are actually several undergoing works in order to precise the notion of platform [Paulo 04] [Delatour 05].

A practice exists for differentiating the platforms which are dependent on a technology (the PSM: Platform Specific Model) and those which are independent (the CIM: Computational Independent Model and the PIM: Platform Independent model). This notion of « Platform independence » is difficult to define and is quite relative. The distinction between a PIM and a PSM is not clear-cut and will be dependent on whether one wants to consider different sets of target platforms. Although often used, these notions of PIM and PSM should be more clarified

2.3 Model transformation

In MDE, the translation from a PIM to a PSM is generally described as a transformation. A transformation consists in generating a set of target models from a set of source models. Target and source models must conform to meta-models. The transformation itself is a model that conforms to a meta-model. There is a huge variety of transformation languages [Czarnecki 03].

One can use traditional languages. For instance, a transformation could be written in Java using a library in order to facilitate the manipulation of the models. Several libraries have been defined in order to manipulate elements from model repositories (JMI for MDR, a MOF compliant repository, EMF for Eclipse framework...).

One can use dedicated transformation languages. There are a lot of different approaches, from declarative or imperative paradigm, based on graph grammar theory, based on template approach... In MDA, the QVT (Query, Views and Transformations) language [QVT 05] has been normalized in order to specify transformations. To the best of our knowledge, there is no implementation of QVT,

currently some transformation languages implement parts of it, but none are totally compliant.

Model transformations are not restricted to the translation of a PIM to a PSM, and a huge variety of transformations could be imagined (from the refinement of a PIM to the code generation). Thanks to the MDE facility for importing and exporting existing models (stored for instance in various textual format or in XML form), it is possible to define a transformation from a specification model to a validation model (for instance, in a petri net formalism). Therefore, it is possible to use the validation tools developped outside the MDE world. For a researcher, it is a gain of time that will be profitably invested in the definition of transformation rules. Indeed, developing its own transformation and parsing technology to different everchanging file format could be a time consuming activity, Moreover, the MDE community is starting to share their meta-models and transformations, available in meta-model zoos and transformation zoos.

3. MDE for Aibo

3.1. Approach

The figure 3.1 presents the MDE approach proposed to build Aibo Software.

Three meta-models have been built (M2 layer):

- The Robot meta-model defines concepts used to specify robots characteristics (joins and sensors). Thanks to this meta-model, models of the different members of the Aibo family can be elaborated. This meta-model has been used to elaborate the model of the ERS-7 member of the Aibo family. The ERS-7 is the last member manufactured by Sony and is the robot we used for running our software. For sake of simplicity, we won't present this meta-model in this paper. The complete meta-model is available in [AiboDev 06].
- The Validation meta-model defines concepts used to specify Aibo consistent and inconsistent states and transitions. For instance, if Aibo is seated, it should not be allowed to walk. This meta-model is used to elaborate different validation models that contain inconsistent states and different strategies (transitions) to gain consistent states. For instance, if Aibo is seated and has to walk, it should stand up first. For sake of simplicity again, we won't present this meta-model in this paper. The complete meta-model is available in [AiboDev 06]
- The Behavior meta-model defines concepts used by developers to specify the Aibo software they want to build. This meta-model can be

considered as the Aibo programming language. This meta-model is detailed in the following section.

In this approach, two kinds of users have been identified. One user is responsible of the elaboration of the robot models and the validation models. This user customizes the MDE environment for Aibo. This user is called the framework level user (M1: framework in figure 1). The other user is the Aibo developer. He/she elaborates models of the behaviors of Aibo. Those models have to be consistent with the robot models and the validation models. For instance, the behavior model cannot target joins and sensors which are not specified in the robot model. The behavior models cannot specify inconsistent behaviors (identified in the validation models).

Once the behavior model is completely elaborated, it can be automatically transformed into code thanks to predefined templates. The code generation step is presented in the following section.

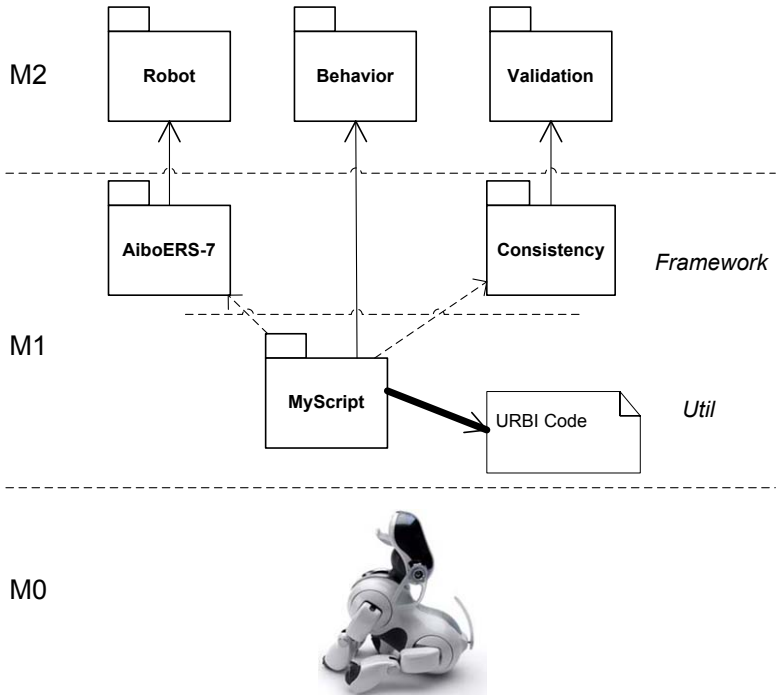

Figure 3.1: MDE Approach for Aibo

3.2. The Aibo meta-model

The figure 3.2 presents a simplification of the Aibo Behavior meta-model. The whole meta-model is available in [AiboDev06].

Basically, this meta-model defines that a Aibo behavior is a kind of sequential script composed of blocs of Aibo actions. Aibo actions are linked to joins and sensors of the Aibo robot model. For instance, the "walk" is specified as a bloc composed of several Aibo actions making Aibo moves its legs in order to walk. Blocs can be controlled by classical control blocs (if, loop, while, etc.). It should be noted that, for sake of simplicity, only the Loop meta-class is presented in figure 2.

Events received by Aibo are considered as condition of the "if" statement. For instance, when a developer wants to specify how Aibo react when it see its ball, he/she has to use the "if" statement with a kind of "ballVisible" condition. Pre-defined event that can be used in condition are specified in the robot models.

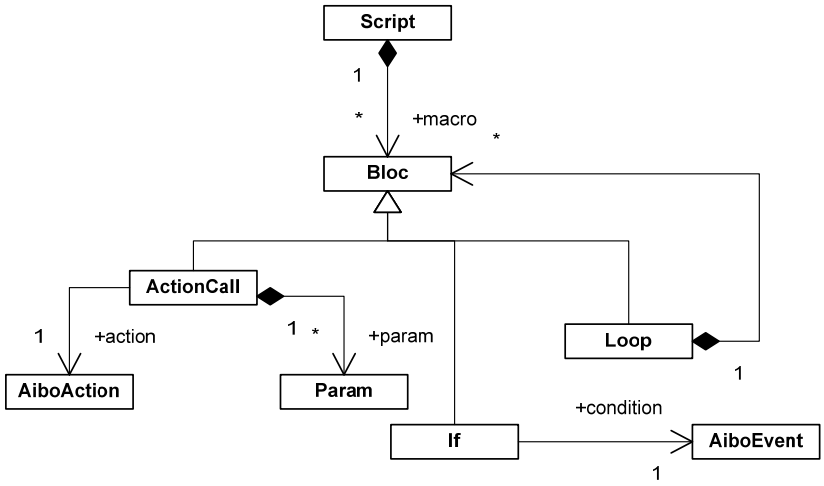

Figure 3.2: Simplification of the Aibo behavior meta-model

In order to validate this meta-model, three behaviors have been considered. Those behaviors are representative enough of behaviors developers usually want to specify:

- **A dance**. There are several dance competitions with Aibo. The dance behavior mainly consists in playing a song and chaining different dance steps.
- **A defense**. Aibo is sometime used as a defense pet. The defense behavior mainly consists in moving in a place looking if somebody moves and then bark.

- **A labyrinth escape**. Aibo can evolve in unknown places. Therefore, it has to know how to escape all places. There are several labyrinth escape algorithms. We choose one consisting in running parallel to walls.

All those behaviors have been completely modeled thanks to our meta-model, see [AiboDev06] for models of those behaviors.

3.3. Aibo to URBI transformation

There are few programming languages targeting the Aibo platform. The best one is probably URBI (Universal Real-time Behavior Interface) [URBI07] which is a scripting language specially dedicated to robots.

Programming with URBI is quite easy as it provides primitives for controlling joins and sensors of the robot. Moreover, URBI provides real-time primitives allowing actions to be performed in specific time frames.

URBI code generation from Aibo models can be seen as a kind of Model-to-Tex transformations (cf. section 2). To realize this transformation, a set of rules have been defined. Each transformation rule concerns a specific meta-class of the Aibo Behaviour meta-model, and targets a specific URBI primitive. The complete set of these transformation rules can be downloaded from [AiboDev 06].

The rules definition is not of particular interest since the Aibo behavior meta-model is script oriented and really fits to URBI. Thanks to them, URBI code can be automatically generated.

The rules have been validated on the three models of the behaviors presented in the last section. For each model a complete URBI code has been generated. This code was then deployed and run into Aibo ERS-7.

4. Conclusions and perspectives

The objectives of our study was (1) to analyze advantages provided by models in this specific robotic context, (2) to measure the maturity of MDE provided technologies and (3) to highlight the limitations of the approach.

In robotic we have used models in order to specify the robot characteristics, the consistent and inconsistent states of the robot and its behaviors. Thanks to the definition of the three corresponding meta-models we were able to specify links between them and to develop automatic operations (validation and code generation). We can say that MDE definitively are useful and offer advantages all specific domains.

Even if we have not presented the realization in this paper, the whole approach has been completely prototyped. EMF (Eclipse Modeling Framework) has been

used to realize editors for each meta-model. Thanks to those editors, user can graphically elaborate models. The validation module has been developed in Java and the code generation with MofScript [MofScript07]. The prototype realized clearly shows that MDE technologies are completely mature.

Even if code can be completely and automatically generated from models, we are not completely sure if it is better to elaborate an Aibo behavior model rather than directly write URBI code. Indeed, our Aibo behavior meta-model is maybe to much script oriented. This is, for us, the clear challenge of applying MDE: define the ideal domain meta-model.

5. Acknowledgment

The authors whish to thank Aleksandra Dworak, François Nizou and Nicolas Ulrich for having defined and realized this approach during the AiboDev competition.

Bibliographie

[AiboDev 06] http://www-src.lip6.fr/homepages/Xavier.Blanc/courses/CAR/AiboDev2006/

[AS MDA 06] AS MDA, « L'ingénierie dirigée par les modèles. Au-delà du MDA », Traité IC2, série Informatique et Systèmes d'Information, Hermes, 2006

[Budinsky 03] Budinsky, F., Steinberg, D., Raymond Ellersick, R., Ed Merks, E., Brodsky, S. A., Grose, T. J., « Eclipse Modeling Framework», Addison Wesley, 2003

[Bezivin 04] Bézivin, J., « In Search of a Basic Principle for Model Driven Engineering ». CEPIS, UPGRADE, The European Journal for the Informatics Professional V(2):21—24, 2004

[Czarnecki 00] Czarnecki, K., Eisenecker, U., « Generative Programming: Methods, Tools, and Applications », Addison-Wesley, 2000

[Czarnecki 03] K. Czarnecki and S. Helsen, « Classification of Model Transformation Approaches » Workshop on Generative Techniques in the Context of Model-Driven Architecture, OOPSLA'03, 2003

[Delatour 05] Delatour J., Thomas F., Savaton G., Faucou S., « Modèle de plate-forme pour l'embarqué », IDM '05, www.plantemde.org/idm05, 2005

[Favre 04] Jean-Marie Favre, « Foundations of Model (Driven) (Reverse) Engineering», several articles (episode 1, 2, 3), www-adele.imag.fr/~jmfavre, 2004

[Frake 97] Franke H., Sztipanovits J., Karsai G., « Model-Integrated Computing », Hawaii Systems of the World Manufacturing Congress, Auckland, New Zealand, 1997

[Greenfield 04] Greenfeld J., Short K. with Cook S., Kent S., (foreword by Crupi J.) « Software Factories, Assembling Applications with Patterns, Models, Frameworks and Tools », Wiley Publishing, 2004

[MDA 03] OMG, MDA guide version 1.1, http://www.omg.org/mda/, June 2003

[MofScript 07] http://www.eclipse.org/gmt/mofscript/

[Paulo 04] Joao Paulo Almeida, Remco Dijkman, Marten van Sinderen et Luis Ferreira Pires, « On The Notion of Abstract Platform in MDA Development », IEEE International Enterprise Distributed Object Computing Conference (EDOC'04), p 253-263, IEEE Computer Society, 2004.

[QVT 05] OMG, « MOF QVT Final Adopted Specification», OMG Document ptc/2005-11-01, http://www.omg.org/docs/ptc/05-11-01.pdf, 2005

[Rothenberg89] Jeff Rothenberg, "The Nature of Modeling.", Artificial Intelligence, Simulation, and Modeling, L.E. William, K.A. Loparo, N.R. Nelson, eds. New York, John Wiley and Sons, Inc., 1989, pp. 75-92.

[Soley 02], Richard Soley and the OMG Staff Strategy Group, « Model Driven Architecture », www.omg.org, 2002

[URBI 07] http://www.urbiforge.com/