MDE between Promises and Challenges

Tahar Gherbi, Djamel Meslati, Isabelle Borne

MDE between Promises and Challenges

Tahar GHERBI
Computer science department
University of Tebessa
Tebessa, Algeria
tahar_gherbi@yahoo.com

Djamel MESLATI
Computer science department
University of Annaba
Annaba, Algeria
meslati_djamel@yahoo.com

Isabelle BORNE
Computer science department
University of Bretagne-Sud
Vannes, France
isabelle.borne@univ-ubs.fr

***Abstract*—Model Driven Engineering (MDE) is a software development approach family based on the use of models in the software construction. It allows the exploitation of models to simulate, estimate, understand, communicate and produce code.**

This paper tries to introduce MDE and presents its actual state. It starts by explaining the industrial world need for such approaches, sheds light on MDE's principles, examines these principles in practice through a relevant alternative named Model Driven Architecture (MDA) of the Object Management Group (OMG), discusses Microsoft's Software factories and IBM's Manifesto MDA versus OMG's MDA and presents some challenges facing the success of MDE.

Keywords-MDE; MDA; PIM; PSM; Software factories

I. INTRODUCTION

Strategies of nowadays companies are mostly based on their computing systems. Functionalities which computing systems offer, their facility of use, their reliability, their performance and their robustness are qualities which let a company to be competitive.

New technologies appear continuously and rapidly. Each decade has known its pack of techniques: symbolic languages, functional programming, structured programming, database management systems, abstract machines, 4th generation languages, object-oriented programming, design patterns, middle-wares (like CORBA, J2EE, .NET), component programming, service approach, etc.

It becomes increasingly difficult for companies to remain competitive. Obviously these technologies simplify the programming and bring gains in the productivity but increase enormously the software complexity; consequently, companies are very embarrassed: adopt a new technology and pay for the expensive migration cost or do not adopt it and take the risk of seeing competitors which make the reverse choice become more competitive.

Model Driven engineering (MDE) comes to improve this situation: its principal aim is "Model once, generate anywhere [11]".

II. MODEL DRIVEN ENGINEERING

Model Driven Engineering, also called Model Driven Development (MDD) or Model Driven Software Engineering (MDSE), is a software development approach family based on the use of models in the software construction. It allows the exploitation of models to simulate, estimate, understand, communicate and produce code.

Bill Gates said: "Modeling is the future, so every company that's working on this I think it's great" [4].

The main idea of MDE is to consider that everything is a model, as in the object approach we considered that everything is an object.

Models (such as analysis and design ones) are mainly used to document software systems. Indeed, often analysts and designers produce models and provide them to programmers as materials of inspiration to facilitate and guide the software production. MDE let us pass to a new situation where production tools will be themselves directed by models; in other words, models can now be directly used to supply automatic software production tools.

The basic ideas of model engineering are closer to those of many other approaches, such as: generative programming, Domain Specific Languages (DSL), Model Integrated Computing (MIC), etc. [1]

Several alternatives exist in MDE; they do not necessarily adopt the same standards, but they share three basic principles: [10]

- Direct representation, by offering families of DSLs, allowing the representation of each situation (aspect) of the system as well as co-operative communities.
- Automation: allowing the mapping, in an automatic way, of models established according to DSLs. This principle treats questions of model integration, model transformation, model weaving, etc.
- The use of open standards to quickly attract researchers, tool suppliers and industrial users.

As examples of MDE's alternatives, we've, among others, Model Driven Architecture (MDA) of the Object Management Group (OMG), Microsoft's Software factories and IBM's MDA Manifesto.

III. MODEL DRIVEN ENGINEERING AND THE OBJECT APPROACH

Note that the object approach has not eliminated the procedural approach; similarly, MDE does not come to replace the object approach: the two approaches can rather

be seen as complementary. Indeed, more precise bridges between models and objects exit, like Java Metadata Interface (JMI) which lets passing from the model world into the Java programming world. [1]

The object technology can, however, be a privileged generation target for the model engineering; but, it is not the only target; other targets can also be candidates, like semi-structured data (as the XML technology), rules, middle-wares, etc.

IV. MODEL DRIVEN ARCHITECTURE

MDA was the first relevant proposal as an MDE approach. It's an initiative of the OMG, published on November 2000. It can be defined as the realization of MDE principles around a set of standards like [1]: Meta Object Facility (MOF), XML Metadata Interchange (XMI), Common Warehouse Meta-model (CWM), Unified Modelling Language (UML), Software Process Engineering Meta-model (SPEM), Query/View/Transformation (QVT), Object Constraint Language (OCL), etc.

Being an MDE alternative, MDA would cover all concerns which can appear during the realization or the maintenance of software: functional aspects (also called business descriptions) and non-functional aspects as: quality of service, reliability, safety, etc. However for the OMG, which was a priority in MDA was the separation between platform dependant aspects and platform independent aspects (neutral business descriptions).

MDA was announced as a cheap solution allowing companies to easily migrate their applications from one platform to another and hence to future platforms also.

Let us see MDE principles in practice through the presentation of MDA.

A. MDA Architecture

MDA uses models in various steps of software development cycle. It recommends the elaboration of:

- Requirement models: Computation Independent Models (CIM),
- Analysis and design models: Platform Independent Models (PIM) ,and
- Code models: Platform Specific Models (PSM).

Models to cover other steps of software development such as checking and testing, supervision, organization, etc., have not been integrated.

Experiments undertaken with MDA have allowed, at least in particular contexts, the validation of the principle of a model driven approach. Then MDA has been enlarged to the wider approach which is MDE [5]. Indeed, MDA has been generalized, under the term MDE, to all model types (not only MOF/UML models, to not limit it to OMG technological advances) in conformity with the "Direct representation" principle of MDE.

1) Global view of MDA: As shown in Fig. 1 [9], the development of a new application starts by making one or more CIM; then PIM are established from CIM; finally, PIM are transformed into PSM.

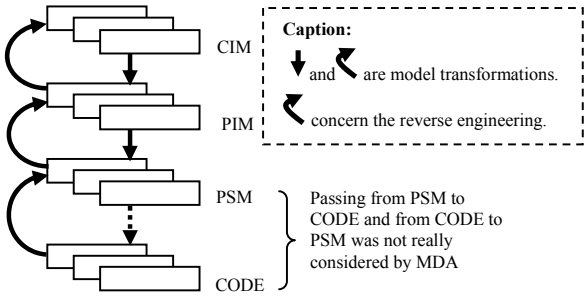

Figure 1. Global view of MDA

MDA recommends also the modelling of model transformations.

2) Meta-modeling and Meta-Meta-modeling: CIM, PIM, PSM, transformation models and any other model which can appear in MDA are expressed, each one, in a particular formalism called meta-model.

MDA recommends the modelling of meta-models as well and proposes the MOF standard as the meta-meta-model.

Fig. 3 [9] summarises the situation.

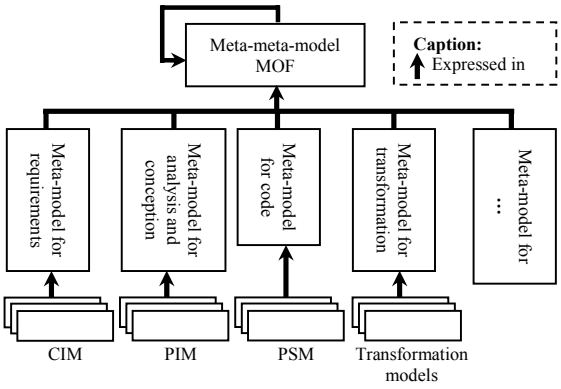

Figure 2. Models, meta-models and meta-meta-model in MDA

B. Software development with MDA

MDA is an approach, not a method. In order to use it, we need to define a method. Thus each enterprise which has practised MDA in its context has defined its own method. These methods respect the MDA approach by using models as productive elements and differ in their automating degree and the set of model operations they propose.

An example of development steps with MDA can be as illustrated by Fig. 3 [2]:

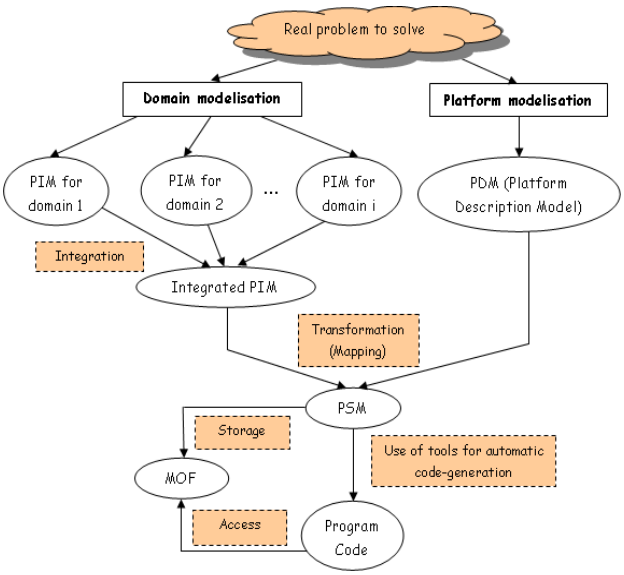

Figure 3. An example of development steps with MDA

Note that:

- If we present the CIM models in this figure, they will appear on top of PIM models.
- In addition any model, not only PSM models, can be stored and accessed using MOF facilities.
- Finally, it's better to model the platforms with PDM, rather than describing a transformation for each target platform. This allows the construction of generic transformations [3].

V. SOFTWARE FACTORIES AND MDA MANIFESTO VERSUS OMG'S MDA

A software has several aspects of various complexities: pure syntactic aspects, business aspects, legacy, safety, quality of service, reliability, etc. During the software construction, a team member may define the global software architecture, while another specifies non-functional aspects related to the safety or required performance, a third team member produces test specifications and so on.

If computer programming languages deal with pure syntactic aspects, the organization of each other aspect concerns a DSL which will be defined. [1]

One criticism of OMG's MDA is that it focussed on one modelling dimension: the dichotomy between PIM and PSM. Kent [7] defines two additional categories of dimensions for MDE:

- The first category concerns the distinction between models:
  - ✓ on the basis of subject area they belong to: different users of a system may have different viewpoints on it, focusing on a subset of system features; and
  - ✓ on the basis of a set of non-functional system aspects, like: data, presentation, concurrency control, security, error handling, etc.
- The second category concerns the organisational issues. It includes authorship, version (as in version and configuration control), location (in case the system development is distributed across sites) and stakeholder (such as business expert and programmer involved).

For any software project, it is necessary to determine its important dimensions. Usually, authorship and versioning are important, however, the points of interest concerning the other dimensions need also to be identified, such as which subject areas play a role, which system aspects are important, which stakeholders are involved, which is the level of abstraction to be used, etc [7].

Thus an MDE approach has to define multiple DSL's, together describing an application. Defining models using these DSLs will result in multiple, connected models defined in different languages, which have to be combined and made executable using a transformation process. [6]

Another criticism of OMG's MDA is that it used a general purpose language for modelling, which is UML. Probably, this may be also the case of other initiatives in the field of MDE. This has the disadvantage that users of the modelling language have to learn all concepts available in it, while, using DSLs presents many advantages: [6]

- The modelling language can be fully tailored to the goal it has to be used for.
- Users work with concepts they know. They only have to learn the limited set of concepts related to the domain they are working on.
- Tools can be tailored to the use of specific DSLs, and can even help the user in defining models with an easy way.

In contrast to OMG's MDA, Microsoft's Software factories are primarily founded on DSLs of small size, which are easy to manipulate, transform, combine, etc. Microsoft will, however, continue to use UML for documentation and also to produce and communicate drafts. The Microsoft's vision for MDE is gradually implemented in Visual Studio. [5]

In the same way, IBM's MDA Manifesto uses DSLs which are adapted to particular corporations or particular needs. UML can also be used within MDA Manifesto but as any other standard, like XML, etc. IBM's vision for MDE is implemented, for example, in Eclipse Modelling Framework (EMF) tools. [5]

VI. MODEL DRIVEN ENGINEERING BETWEEN PROMISES AND CHALLENGES

The real motivation for MDE is to preserve analysis, conception and development efforts, improve productivity and facilitate the migration of applications from one platform to another. Indeed, in the current complex enterprise world, everyday new technologies are evolving and enterprises have to keep up-to-date with them. Requirements from customers are changing and functionalities have to be added to the existing software,

which may require new technologies. Therefore MDE aims to enable the integration of the existing system with the new system and any other system that may be created in the future.

MDE is still a high level vision and is not worked out in details. If we want to start building softwares in an MDE way, we will need to look for some methodologies based on ideas and practical experiences from others.

The complexity of software systems has increased much faster than the productivity software development; consequently, to reach its promises, an MDE approach, has to pay attention to many challenges, among them [6, 8]:

- Change in business and developer personnel: vital knowledge should be expressed in a concise and tailored presentation understandable by all involved people.
- Change in customer requirements.
- Independence of development platforms.
- Independence of deployment platforms.
- Model change propagation: if changes occur in a source model, they must propagate into the target model.
- Ease in the definition of model transformations.
- Model testing and debugging tools.

Actually, MDE is a big undertaken framework; therefore, we cannot confirm yet if it will solve all software development problems; however, all opinions confirm that this direction is very promising.

VII. Conclusion

MDE is gaining more and more interest. It was first mentioned by Kent [7] and is wider than MDA: it defines multiple modelling dimensions in addition to the platform dependent/independent dimension defined in MDA.

The first experiments undertaken with MDA have allowed, at least in particular contexts, the validation of the principle of a model driven approach; then MDA has been enlarged to MDE.

An MDE Approach handles many formalisms belonging to various technological domains and located at various abstraction levels. Consequently many questions arise, such as “How conceivers will adapt themselves to this logic?”, “How the complexity resulting from it can be controlled and masked by tools?” etc.

Several researchers and industrialists agree to think that MDE is the future of applications and thus are involved actively in its evolution. In fact, MDE can be seen as a family of approaches which are developed simultaneously in the research laboratories and in the industrial companies which led to great software development projects.

This paper aimed to present MDE, its principles, its promises and challenges and to situate within it one of its relevant alternatives: the OMG’s MDA. This study is a first step for our project under development, concerning the modelling and implementation of mobile agent applications with a MDE approach.

References

[1] J. Bézivin, “Sur les principes de base de l'ingénierie des modèles,” RSTI l'Objet, special number “Où en sont les objets?,” vol.10, 2004, pp.145-156

[2] P. Parrend “Introduction à MDA : Principe,” 2006, available: http://pparrend.developpez.com/tutoriel/mda-intro/

[3] F. Thomas, J. Delatour, S. Gérard, M. Burn and F. Terrier, “Contribution à la modélisation explicite des plates-formes d'exécution pour l'IDM,” Revue l’OBJET, article Vol 13/4, 2007, pp.9-31

[4] “What Is Bill Gates Thinking?,” Interview, eWEEK.com, 3/30/2004

[5] J. Bézivin, M. Blay, M. Bouzhegoub, J. Estublier, J.M. Favre, S. Gérard and J. M. Jézéquel, “Rapport de Synthèse de l’AS CNRS sur le MDA (Model Driven Architecture), ” 2004, available: http://www.cnrs.fr/

[6] J.D. Hann, “8 Reasons Why Model-Driven Approaches (will) Fail,” Jul. 2008, available: http://www.theenterprisearchitect.eu/archive/2008/07/29/

[7] S. Kent, “Model Driven Engineering,” Proc IFM2002, LNCS 2335, 2002, pp. 286-298

[8] C. Atkinson, T. Kühne, “Model-Driven Development: A Metamodeling Foundation,” IEEE Softw., vol. 20, no. 5, 2002, pp. 36-41

[9] X. Blanc, MDA in action : Ingénierie logicielle guidée par les modèles, Ed. Eyrolles, Mar. 2005

[10] G. Booch, A. Brown, S. Iyengar, J. Rumbaugh, B. Selic, “An MDA Manifesto,” MDA Journal, May 2004, available: http://www.bptrends.com

[11] J. Bézivin, “Les nouveaux défis des systèmes complexes et la réponse MDA de l’OMG,” JFIADSMA'02 (Lille, France), Oct. 2002, available: http://www2.lifl.fr/jfiadsma2002/talks/