MDE and Mobile Agents : another reflexion on the agent migration

Tahar Gherbi, Isabelle Borne, Djamel Meslati

MDE and Mobile agents: another reflection on the agent migration

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

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

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

***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.**

Mobile agents are a very interesting technology to develop applications for mobile and distributed environments. A mobile agent is essentially a computer program that acts autonomously on behalf of a user and travels through a network of heterogeneous computers. By moving to remote computers, the mobile agent processes data there and hence saves remote communications and local computation.

Many mobile- agent platforms have been developed. While some of them have been abandoned, others continue to release new versions that fix bugs detected or offer new interesting features.

In this paper, we present a reflection on how one can realize the migration operation when the mobile-agent applications are built using an MDE approach.

Keywords-MDE; PIM; PSM; mobile agent; agent factory; agent migration

I. INTRODUCTION

Model Driven Engineering (MDE) is gaining more and more interest. Indeed, new technologies (design patterns, middle-wares, component programming, service approach, etc.) appear continuously and rapidly; then, it becomes increasingly difficult for companies to remain competitive. Obviously these technologies simplify the programming task and bring gains to the productivity but enormously increase 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.

MDE comes to improve this situation: its principal aim is "Model once, generate anywhere [16]".

Several researchers and industrials agree to think that MDE is the future of applications and are actively involved in its evolution. It is thus reasonable to expect that the coming years will know a rich set of operational tools and working platforms supporting MDE.

In this paper we consider the mobile-agent platforms and more precisely the "migration" operation which moves an agent from one computer to another. We present a reflection on how this operation can be realized if the mobile-agent applications are built using an MDE approach.

The rest of the paper is organized as follows: Section 2 introduces MDE. Section 3 presents the mobile-agent technology, describes the migration operation and examines an approach (the generative migration using an agent factory [16]) supporting the agent migration over heterogeneous mobile-agent platforms. Section 4 discusses a reflection for an MDE based vision to realize the migration operation. Section 5 presents our conclusions.

II. MODEL DRIVEN ENGINEERING

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.

Several alternatives exist in MDE; they do not necessarily adopt the same standards, but they share three basic principles: [15]

- 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 have, among others, Model Driven Architecture (MDA) of the Object Management Group (OMG), Microsoft's Software factories and IBM's MDA Manifesto.

Let us see MDE principles in practice through the presentation of the OMG's MDA which was the first relevant proposal as an MDE approach.

A. *MDA Architecture*

MDA is an initiative of the OMG, published on November 2000. It can be defined as the realization of MDE principles around a set of standards, such as [1]: Meta Object Facility (MOF), XML Metadata Interchange (XMI), Common Warehouse Meta-model (CWM), Unified Modeling 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 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 [14], 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.

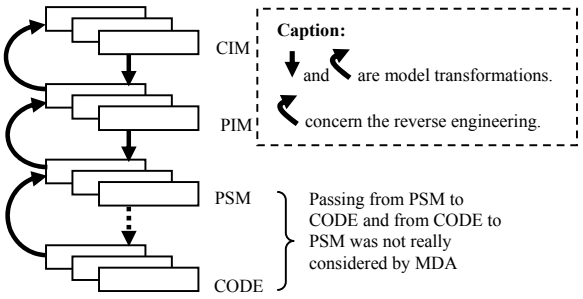

Figure 1. Global view of MDA

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

Fig. 3 [14] summarizes the situation.

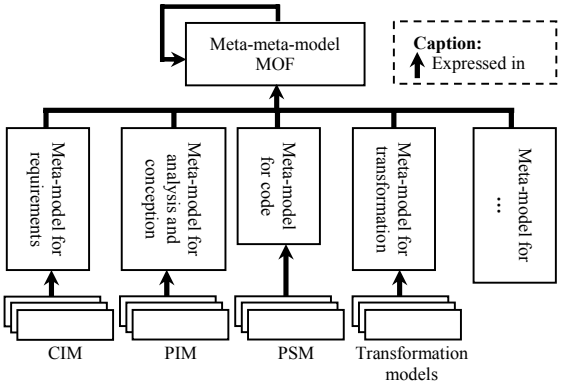

Figure 2. Models, meta-models and meta-meta-model in MDA

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 practiced 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]:

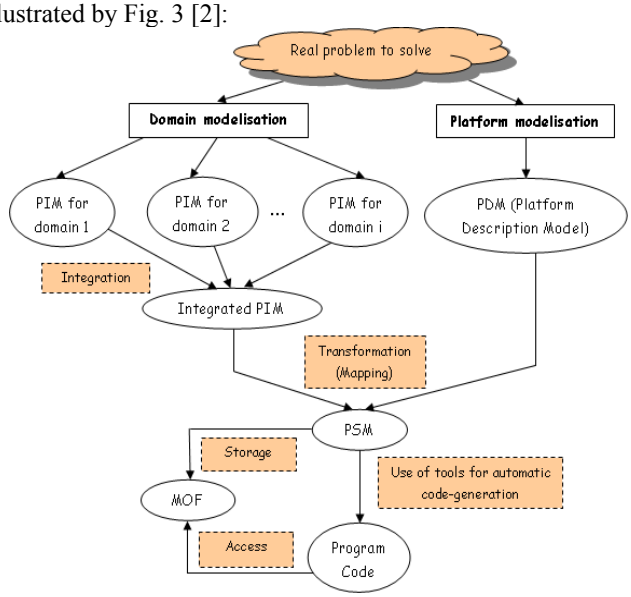

Figure 3. An example of development steps with MDA

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].

III. MOBILE- AGENT TECHNOLOGY

A mobile agent is essentially a computer program that acts autonomously on behalf of a user and travels through a network of heterogeneous computers. It executes within a mobile-agent platform. This later must offer a set of agent services, such as [7]: creation and designation, access to local resources, communications, migration between computers, agent localization and traceability, agent protection from visited computers and other agents, computer protection against visiting agents, fault tolerance, etc.

A. Mobile agent structure

As a mobile agent is a computer program, it has then an execution flow defining its tasks; and because it can migrate from one computer to another, it is important to know which elements must be transferred with it.

An instance of mobile agent contains three parts [7]:

- An executable code representing the mobile-agent static comportment,
- An execution context (register values, execution stack) which reflects the mobile-agent current execution state, and
- Resources which the mobile agent uses. These resources are of two kinds [8]:
  - ✓ Transferable resources which are the agent attribute values forming its global state,
  - ✓ Non transferable resources which form the execution environment offered by the mobile-agent platform (for example, opened files, socket connections, etc.).

B. Mobile agent migration

This operation, as illustrated in Fig. 4, realizes the transfer of an agent in execution from one computer to another through a network. When it migrates, the agent takes with it: its code, its execution context and/or its transferable resources.

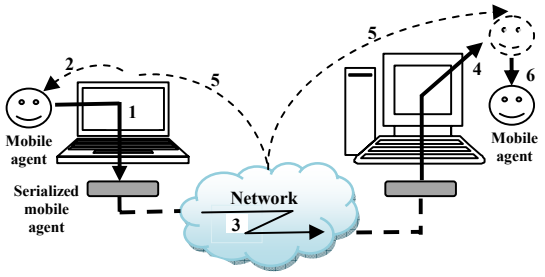

Figure 4. Mobile agent migration

The migration operation executes the following steps [9]:

1. Suspends the agent execution and serializes its necessary details: code, execution context and/or transferable resources.
2. Suspends all communications with other agents and the agent dies.
3. Sends the serialized agent details to the destination computer.
4. Restores the agent on the destination computer.
5. Redirects suspended agent communications towards the destination computer. Note that the agent was restored but is not yet activated.
6. Activates the agent. The migration operation ends and any saved information concerning it, can be definitely erased from the source computer.

C. Migration strategies

Two types of migration have been proposed in the existing mobile-agent platforms [10]: strong and weak migration.

1) Strong migration: The agent can migrate at any moment regardless its execution state. After its migration, the agent continues its execution from the point where it stopped last.

The strong migration needs a mechanism which instantly captures the agent execution state. Then the agent's code, its execution context and its transferable resources are serialized and transferred.

Mobile Object Reactive Agents (Moorea) is an example of an agent platform supporting the strong migration. [7]

2) Weak migration: In the weak migration, the agent takes with it only its code and its transferable resources. On the destination computer, the agent re-executes from the beginning and its execution context is re-initialized.

The weak migration operation can either be called at any moment during the agent execution (but in this case we lost the agent work), or it can be called at privileged moments (stop-points) which need to be defined in the agent's code.

As an example of an agent platform supporting the weak migration, we have Jade [13].

D. Migration scenarios

The sites involved in a distributed mobile- agent application (for example, an application searching for

information on many different sites on the Internet) may not contain the same agent platform or not have the same (virtual) machine. In addition, the executable code of an agent usually contains a part that provides the interface between the agent and the agent platform on which it runs. Hence, several migration scenarios, as summarized in Table 1, result: [11]

TABLE 1. MIGRATION SCENARIOS

	Mobile- agent platforms on the source and target sites	**(virtual) machines on the source and target sites**
Homogeneous migration	same	Same
Cross-platform migration	different	Same
Agent-regeneration migration	same	different
Heterogeneous migration	different	different

1) Homogeneous migration: In this scenario, an agent migrates to another site without any change either to the format of its executable code or to its interface to the agent platform. In practice, this form of migration is most common.

A homogeneous migration can be either strong or weak. Very few agent platforms support the strong migration; it requires that all sites must have exactly the same operating system, the same machine architecture and the same (version of the) programming language. The most of popular agent platforms are Java-based and support the weak migration. [10]

2) Cross-platform migration: In this scenario, an agent migrates to another site having a different agent platform but running on the same (virtual) machine. This generally implies changes on the agent platform interface but not necessarily changes to the format of the agent's executable code. [11]

Many solutions have been proposed, such as: [11]

- Constructing wrapper interfaces which hide differences between source and target platforms, or
- Constructing agent platforms which are compliant to a standard interface for interoperability. As such standards, we have the Mobile Agent System Interoperability Facility (MASIF) of the OMG and the Foundation for Intelligent Physical Agents (FIPA).

3) Agent-regeneration migration: In this scenario, an agent migrates to a site having the same agent platform but running on a different (virtual) machine. This requires the regeneration of the agent, resulting in a different executable code.

4) Heterogeneous migration: An agent migrates to another site having a different agent platform and running on a different (virtual) machine. In this case, the agent regeneration is also necessary.

E. Generative migration using an agent factory

Interesting initiatives for the agent migration across different platforms exist, such as proposed in Guest [13], Monads [1] and the generative migration using an agent factory [11]. Unfortunately, no source-code or libraries on Guest or Monads are available for the public or the research usage [10].

Concerning the generative migration using an agent factory, it aims to support all the migration scenarios presented in the section 3.4 and especially the heterogeneous migration.

The concept of an agent factory requires: [11]

1. The agent to have a compositional structure (the resulting specification is called blueprint),
2. One or more libraries of re-usable agent components, and
3. One or more ways to describe the functionality of these agent components.

An agent factory is a service capable of: [10]

1. (Re-) design (i.e., create a new blueprint) and adapt an agent (that it can, for example, execute in another agent platform),
2. Finding appropriate building libraries, and
3. Assembling and regenerating an agent adapted to its new environment at its destination site.

In short, an agent factory constructs new agents and/or modifies existing ones [12]. The (re-)design of agents is fully automated with very limited interactions with outside parties [11].

The generative-migration approach using an agent factory, considers that an agent has a blueprint which describes its structure and functionality and a state which represents its execution. The agent's state includes its transferable resources and eventually its execution context (in the case the strong migration is supported and solicited).

The key idea of this approach is, rather than sending the agent's code, we send an agent description in an implementation (agent platform and machine) independent format. Both the agent's blueprint and its state are expressed in this format (as a candidate format, we have the eXtended Markup Language: XML).

It's to be expected that the transmission of an agent's blueprint and information on its current state will generally require less network resources than migrating an agent using more traditional approaches. [11]

The blueprint and state descriptions are sent to the destination site, where an agent factory processes them. This later:

- Re-builds the agent from its received blueprint. The appropriate building blocks (libraries) must be available at the agent platform to enable the regeneration of the agent's code. These building blocks are retrieved from a local repository (for the reusable building blocks) and the agent is assembled.

[1] http://www.cs.helsinki.fi/research/monads/

- Initiates the regenerated agent with its state and launches it into the platform where it continues its execution.

IV. MDE FOR ANOTHER REFLECTION ON THE AGENT MIGRATION

MDE aims to mask the heterogeneity of development and deployment environments during the software construction. It consists of elaborating platform independent models, on which a set of model transformations is operated until obtaining the executable code.

We distinguish two types of model transformations in MDE: [2]

- Horizontal transformations: they preserve the abstraction level of the model and are used by concern of:
  - ✓ Representation: It is the question of passing from a format to another. The target format being rich in handling and representation tools, or for which tools are simply available,
  - ✓ Optimization, to improve the performance,
  - ✓ Rebuilding, to improve the maintenance and the visibility of the code.
- Vertical transformations: they change the abstraction level of the model. We distinguish:
  - ✓ Refinement mapping: it is used for example to pass from an analysis/conceptual model to an implementation model (as from PIM to PSM).
  - ✓ Abstraction mapping: it is used for the reverse engineering or for migrate an application from one platform to another.

Several researchers and industrials agree to think that MDE is the future of applications and are involved actively in its evolution. If MDE successes to meet its promises ("modeling once, generating everywhere", hiding the software complexity and preserving their continuity and evolution from obsolete technologies, etc.), we will expect to see MDE supports on the most sites. Consequently, it will be very interesting and even recommended to construct mobile-agent applications with an MDE approach and to elaborate mobile-agent platforms which exploit the MDE supports. Indeed, the agent applications will benefit from the MDE's advantages and the agent platforms will easily support all the migration scenarios presented in the section 3.4.

Let us suppose a mobile- agent application built with an MDE approach, a practical example for an MDA/MDE alternative supporting the weak agent migration (as it is the most common migration type) may be as follows:

1. Any running agent has its code generated from its PIM. This later is saved in a repository using MOF facilities.
2. During the agent execution, if the migration operation is invoked, the agent's transferable resources are gathered and made with the agent's PIM in an independent format (such as XML, using XMI facilities) message.
3. The resulting message is then sent to the target site.
4. On the target site, the agent's PIM and its transferable resource are extracted from the received message. The agent code is rebuilt from its PIM using appropriate transformations (horizontal transformations may eventually apply, for example when vertical transformations do not exist for the received PIM's format).
5. The agent is then activated and its execution context is initialized.

One of the biggest challenges of MDE is to offer tools for testing and debugging the software artifacts at the model level [6]. Otherwise, for the strong migration case, if one can elaborate a mechanism which associates to the execution pointer on the agent's code (pointing its actual instruction) another execution pointer on the agent's PIM, so that the two pointers evolve in a coordinated manner during the agent execution; the pointer on the agent's PIM can thus be sent (within the agent's execution context at model level) to the target site to help restoring the execution of the rebuilt agent at the same instruction at which it was before its migration.

The ideas expressed in the "generative migration using an agent factory" approach [10] can be made closer to those of MDE. Indeed, to "blueprints" can correspond "PIM models", "agent generation" can be replaced by "model transformations", "repository for saving blueprints" can be made MOF compliant, saving and transferring models in XML format can be based on XMI facilities. Hence, the experiences with the implementation of the "generative migration using an agent factory" can effectively help to elaborate PIMs for the mobile-agent application conception, a Platform Description Model (PDM) for each combination of mobile-agent platform and (virtual) machine, model transformations enriching the PIM with the PDM's details and translating the resulting model into PSM. In addition, the corresponding meta-models can be elaborated: the PIM's meta-models can be inspired from the blueprint's languages, the meta-models for PDM and the model transformations can be inspired from the agent factory algorithm. To resume, the ideas expressed in this approach and others dealing with the question of agent migration across different agent platforms may concur to elaborate an MDE alternative for constructing mobile-agent applications and to build agent platforms exploiting the MDE supports. This make the agent applications benefit from the MDE advantages and the agent platforms to support the agent migration by only exploiting the MDE supports existing on each site.

V. CONCLUSION

MDE is gaining more and more interest. It is actually a big undertaken framework; therefore, we cannot confirm yet if it will really solve all the software development problems;

however, all the opinions confirm that this direction is very promising. Moreover, several researchers and industrials agree to think that MDE is the future of applications and are actively involved in its evolution.

MDE is making changes in all the software production industrial practices. This paper has presented a reflection on how we can support the agent migration if the agent applications are built with an MDE approach and the agent platforms can exploit the MDE supports. This allows the agent applications to benefit from the MDE advantages and allows the agent platforms to support the agent migration by just exploiting the MDE supports existing on each site.

We have examined an interesting approach for the agent migration (the generative migration using an agent factory [11]) and concluded that the ideas expressed in this approach and others dealing with the question of agent migration across different platforms may concur to concretize our reflection.

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/mdaintro/

[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," July 2008, available:
http://www.theenterprisearchitect.eu/archive/2008/07/29/

[7] S. El Falou, "Programmation répartie, optimisation par agent mobile," Doctorate's thesis, university of Caen, France, Nov. 2006

[8] A. Fuggetta, G. P. Picco and G. Vigna, "Understanding Code Mobility," IEEE Transactions on Software Engineering, 1998, pp.342-361

[9] L. Ismail, D. Hagimont and J. Mossière, "Evaluation of the Mobile Agents Technology: Comparison with the Client/Server Paradigm," Journal Technique et Science Informatiques (TSI'2000), France vol.19, no.9, 2000

[10] D. R. A. De Groot, F. M. T. Brazier and B. J. Overeinder, "Cross-Platform Generative Agent Migration," Master's thesis, Vrije Universiteit, Amsterdam, Jan.2004

[11] F.M.T. Brazier, B.J. Overeinder, M. van Steen and N.J.E. Wijngaards "Agent Factory: Generative Migration of Mobile Agents in Heterogeneous Environments," in: Proceedings of the 2002 ACM (SAC), 2002, pp. 101-106.

[12] F.M.T. Brazier and N.J.E. Wijngaards, "Automated servicing of agents," In Proceedings of the AISB-01 Symposium on Adaptive Agents and Multi-Agent Systems, March 2001, pp. 54–64

[13] Magnin, L., Snoussi, H., Pham, V. T., Dury, A. and J.-Y. Nie. "Agents Need to Become Welcome," In Proceedings of the 3rd International Symposium on Multi-Agent Systems, Large Complex Systems, and E-Businesses (MALCEB'2002), Germany, Oct. 2002.

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

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

[16] 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/