MDE Adoption in Industry: Challenges and Success Criteria

Parastoo Mohagheghi[1], Miguel A. Fernandez[2], Juan A. Martell[2], Mathias Fritzsche[3] and Wasif Gilani[3]

[1] SINTEF, P.O.Box 124-Blindern, N-0314 Oslo, Norway
parastoo.mohagheghi@sintef.no
[2] Telefónica Research & Development, Valladolid, Spain
mafg@tid.es, jamartell@gfi-info.com
[3] SAP Research CEC Belfast, United Kingdom
{mathias.fritzsche, wasif.gilani}@sap.com

Abstract. Model-Driven Engineering has been promoted for some time as the solution for the main problem software industry is facing, i.e. complexity of software development, by raising the abstraction level and introducing more automation in the process. The promises are many; among them improved software quality by increased traceability between artifacts, early defect detection, reducing manual and error-prone work and including knowledge in generators. However, in our opinion MDE is still in the early adoption phase and to be successfully adopted by industry, it must prove its superiority over other development paradigms and be supported by a rich ecosystem of stable, compatible and standardized tools. It should also not introduce more complexity than it removes. The subject of this paper is the challenges in MDE adoption from our experience of using MDE in real and research projects, where MDE has potential for success and what the key success criteria are.

Keywords: Model-driven engineering, challenges, domain-specific modeling, performance engineering, traceability.

1 Introduction

Today's software systems are complex in nature; the size has been growing because of the increased functionality, heterogeneity is also becoming a bigger concern as systems are built from several systems or include legacy code, systems are distributed over multiple sites and there are new requirements such as dynamicity and autonomy (self-* properties, for example self-healing). Handling each of these challenges requires specific approaches which often include domain-specific knowledge and solutions. However, based on the experience gained from multiple domains and projects, some solutions may be identified as beneficial to complex software development in general.

Model-Driven Engineering (MDE) is an approach built upon many of the successful techniques applied in software engineering: It can be characterized by: a) raising the abstraction level by hiding platform-specific details ; b) taking advantage

of models in all the phases of software development to improve understanding; c) developing specific languages and frameworks to achieve domain appropriateness; and d) taking advantage of transformations to automate repetitive work and improve software quality [1]. These are all techniques useful for complex system development and therefore one may expect rapid adoption of the paradigm by industry. So far, we cannot see such wide adoption, as also confirmed by a review of industrial experiences presented in [2]. To be accepted by the majority, the industry must gain confidence on the promises of MDE and have access to proper tools and experts.

The European research projects MODELWARE[1] and its continuation MODELPLEX[2] have focused on MDE approaches and tools with the goal of making them suitable for complex system development. Some of the companies involved in these projects have experience from applying MDE in real projects while others think that MDE is not yet mature enough to be taken from research projects to industry production. This paper therefore elaborates on where we can expect added value from MDE and what the barriers are from experiences gained in the context of these projects. In the remainder of this paper we discuss industry expectations and experience in Sections 2 and 3 and conclude our discussion in Section 4.

2 SAP Experience

SAP has already started working towards applying MDE concepts, and currently employs models in various stages of business application development. The Composition Environment is one example where MDE concepts are applied for efficient development of Composite Applications. Composite Applications are self-contained applications that combine loosely coupled services (including third party services) with their own business logic, and thereby provide user centric front-end processes that transcend functional boundaries, and are completely independent from the underlying architecture, implementation and software lifecycle. With Composition Environment even the non-technical users, such as business domain experts, consultants, etc., having no programming skills, are able to model and deploy customized applications suited to their specific business requirements.

Based on our experiences with the currently employed tools for MDE of business processes, such as the Composition Environment, we identified the general need of supporting non-technical users with regards to non-functional requirements, such as the impact of their design decisions on performance, etc. Within the context of performance engineering, for instance, such a support means guidance towards better design / configuration that actually meets the timelines, and optimized resource mapping against each activity in the business process.

We implemented such performance related decision support as an extension of MDE. By implementing this extension, named Model-Driven Performance Engineering (MDPE), we realized the need for supporting requirements with respect to non-functional aspects, especially performance. The implementation of MDPE heavily uses the MDE concepts such as meta-modeling, transformations, model

[1] http://www.modelware-ist.org/
[2] http://www.modelplex-ist.org/

weaving and mega-modeling. For instance, ten different meta-modeling languages are employed in order to make the process usable for a number of domain-specific modeling languages. During the implementation of MDPE, we recognized that the application of MDE concepts enabled us to focus on the creative tasks of development rather than repetitive coding. For instance, code generation for our meta-models saved us significant development effort. The only place where a significant amount of coding effort was required was for the integration of MDPE into the existing tool infrastructure.

Meta-model extension is the generally employed technique for model annotations, such as done with profiles in the case of UML [4]. However, this is not applicable while dealing with the proprietary models. The application of model weaving enabled us a high degree of flexibility as we are able to annotate any kind of proprietary model with the help of a generic editor [4]. Higher-order transformations are used to enable traceability in our approach [5]. Additionally, mega-modeling enables us to locate our model artifacts, such as the tracing models related to the models in our transformation chain [6].

As for the challenges, we experienced that MDE concepts are on the one hand very systematic and efficient, but on the other hand also difficult to understand for developers as they require quite a high level of abstraction and training. Also, the MDE tool support is sometimes not mature enough. Especially the available tooling to define model transformation chains lacks capabilities of modern IDEs, which could decrease the development time for model transformations significantly.

Concluding, based on the experiences gained with the development of MDPE, we are optimistic regarding the capabilities of MDE in case the tool support improves, and the MDE community meets the challenges associated with the MDE process, such as providing support for dealing with non-functional aspects of system development.

3 Telefónica Experience

In [3], we have discussed the experience of Telefónica in moving from a code-centric to a model-centric software development. Earlier efforts in modeling failed due to the complexity of UML, the lack of proper tools and the inability to maintain models in synch with code, among other issues. Due to the above problems with UML, we decided to develop our own programming tools and frameworks addressing the problem domain. But without any industry standards to rely on, this approach had no future in the long term and was also difficult to use for non-technical stuff, such as telecom domain experts, as it did not have the required abstraction level.

This was an experience from eight years ago, but not so many things seem to have fundamentally changed. What we look for is a domain-specific modeling (DSM) language integrated in a development environment that will permit the modeling of our basic domain concepts, such as interfaces, devices, networks, protocols and services. We also emphasize adhering to current industry standards in the domain, since we now look for a domain-specific solution, not a company-wide solution. Other requirements are: a) the ability to model in multiple abstraction levels, hiding

details as desired; b) the integration of model verification tools based on OCL or other constraint languages and c) the composition / weaving of the models at run time to reflect the changes in the network's operational status.

In the road toward these objectives we foresee numerous challenges. First of all, the UML standard has evolved but, with this evolution, the syntax has become even more complex and the necessary supporting mechanisms and tools for dealing with this added complexity are not yet available. Even something as conceptually simple as exporting a UML diagram from one tool to another has not been accomplished yet with ease. On the other hand, developing a DSM solution requires high skills related to meta-modeling and tool development. Also a big concern with Domain-Specific Languages (DSLs) is getting the people in that domain to agree upon a standard syntax. Another challenge is having that DSL interact properly with anything outside of its domain, having a different underlying syntax to that of other languages.

Model synchronization (for example applying multiple profiles to a source model) and roundtrip engineering are yet to be addressed successfully and mechanisms for dealing with very large and complex models, such as hierarchical models, traceability and model management in general are also in an inception phase right now, at least regarding to the aspect of tool support. All these features are important to make a full fledged MDE process work in complex, real-life projects.

Another challenge for organizations wanting to get started in MDE, closely related with the previous idea of managing all these artifacts, is that they may end up dealing with more complexity than anticipated at first. The underlying problem here is: are the techniques for handling complexity in danger of making the software engineering process itself too complex? To adequately address complexity we have to substitute it for something simpler not for something different but equally complex.

It is our opinion also that there are some basic milestones a new technology has to go through for it to be considered mainstream. To start with, we need a proper context for it to flourish and be nurtured in. The fabric of this context is made of the proper professionals with the proper knowledge and expertise and supporting material which helps in turn to create these professionals. This has to be accompanied by the development of high-quality literature, tutorials and proper material to draw new professionals in.

The main question that an organization has to ask itself is "do I really need MDE?" The second question relates with its ability to adapt its processes to the ones needed from an MDE point of view (partially discussed also in [3]), adapt their staff to new ways of looking at problems and create new layers of software development supporting all the aspects MDE has to offer. Companies may be reluctant to change either their structure or part of it. Apart from software factories for product line engineering (PLE) we have not identified very good candidates for MDE to be applied to.

4 Conclusions

MDE is a long-term investment and needs customization of environment, tools and processes, and training. For companies that have a product line, MDE can pay off

since this cost is amortized over several projects. For one-of–a-kind projects this will not pay in most cases. Despite differences in domain and the type of systems developed in the two companies, there are common challenges as described here. The most important one is the complexity of developing an MDE environment tailored to the company needs. This environment requires:

- Developing proper languages for communication between technical and non-technical experts and for modeling various aspects. The major challenge here is to have the required language engineering expertise since creating own profiles or meta-models are difficult and for complex systems we probably need several languages. Hence more domain-specific meta-models and profiles are needed that are supported by tools and may be reused. The current tools for developing meta-models and editors are not user friendly, the learning curve is steep and the documentation and support is not satisfactory.
- Several tools are required for modeling, model-to-model and model-to-text transformation, verification and simulation, and other tools to store, reuse and compose models. There is no tool chain at the moment and companies must integrate several tools and perform adaptation themselves.

Both of the above requirements put a high burden on companies that traditionally used third-party tools for modeling and performed programming by hand. Training is another major challenge here. We see advantages in gradual introduction and support by management, as well as in the creation of teams of experts that can give support and create the necessary tools for MDE adoption in the whole company.

Acknowledgments. Part of the ideas presented in this paper are based on conclusions obtained in the MODELPLEX project (IST-FP6-2006 Contract No. 34081), co-funded by the European Commission as part of the 6th Framework Program.

References

1. Mohagheghi, P.: Evaluating Software Development Methodologies based on their Practices and Promises. Accepted at the 7th Int'l Conference on Software Methodologies, Tools and Techniques (Somet'08)
2. Mohagheghi, P., Dehlen, V.: Where is the Proof? A Review of Experiences from Applying MDE in Industry. In ECMDA-FA 2008, LNCS 5095, Springer, pp. 432—443 (2008)
3. Fernandez, M.: From Code to Models: Past, Present and Future of MDE Adoption in Telefónica. In: 3rd Europen Workshop From Code Centric to Model Centric Software Engineering: Practices, Implications and Return on Investment (C2M), co-located with ECMDA 2008, pp. 41—51 (2008)
4. Fritzsche M., Johannes J., et al: Systematic Usage of Embedded Modelling Languages in Model Transformation Chains. Accepted at the Software Language Engineering Conference (SLE'08)
5. Fritzsche, M., Johannes, J., Zschaler, S., Zherebtsov, A., Terekhov, A.: Application of Tracing Techniques in Model-Driven Performance Engineering. In: ECMDA-FA 4th Workshop on Traceability (2008)
6. Barbero, F. Jouault, J. Bezivin: Model Driven Management of Complex Systems: Implementing the Macroscope's Vision. In: 15th ECBS'08, IEEE Press, pp. 277--286 (2008)