European Journal of Information Systems (2007) 16, 346–348

www.palgrave-journals.com/ejis

GUEST EDITORIAL

Model-driven systems development: an introduction

Mark Lycett[1], Esperanza Marcos[2] and Veda Storey[3]

[1]Brunel University, Uxbridge, U.K.; [2]Rey Juan Carlos University, Madrid, Spain; [3]Georgia State University, Atlanta, U.S.A.

European Journal of Information Systems (2007) **16,** 346–348. doi:10.1057/palgrave.ejis.3000684

There is a significant disconnect between the ideals of 'on-demand' business and the way in which we currently develop information systems. Growing competition, globalization, increasing regulation, mergers and acquisitions, and innovative business models all require organizations to be flexible and adaptable. The ability of existing computer-based information systems to meet the challenges of on-demand business is, however, limited.

Historically, business models have been embedded in code that does not distinguish those models from the assumptions and platform constraints of particular technology approaches. In many cases, models are not semantically well formed: The description of business things and relationships vary greatly across different systems and the code is, in fact, the only representation of the model. In addition, in many legacy environments, the human 'memory' of the business knowledge has evaporated as staff members retire and/or move to other jobs. The reality of this situation is evidenced in the general cost profile of systems development, where the bulk of spending and effort lies in the (evolutionary) maintenance, integration and interoperability of existing systems – not in the development of new systems. This situation provides a challenge to the generally accepted approaches to systems development.

In response, model-driven development (MDD) (Selic, 2003; Bézivin, 2005) has emerged as a potential means of divorcing business issues from the underlying technology platforms, in a way that makes change more manageable. Model-driven approaches see the primary system development artefacts only as models and their transformations. The technological element of an information system is simply generated from models. Models have always been considered in software development and, as the technologies related with software development have evolved, the role models play in software development has taken on great relevance. Until MDD appeared, models were generally considered just as simple documentation and, in the best case, they were used to generate a reduced skeleton of the final code. From this point of view, models were discarded as soon as the corresponding development phase was finished, and they were not updated to reflect the changes made in subsequent models or in the working code. Consequently, those models could not be used for maintenance or management tasks as they did not accurately reflect the system deployed. With MDD, this situation has changed – automation comes as the other key element. Developers therefore have an opportunity to shift their focus from coding to modelling, defining models with as much accuracy as possible in order to capture all the requirements and specifications of the system alongside the platform where it will be deployed. In theory, the final code for the whole system (and not only a skeleton) is then automatically generated from these models.

The model-driven architecture

Following the principles of MDD, the Object Management Group (OMG) proposed, in 2001, the Model-Driven Architecture (MDA) (Kleppe *et al*., 2003; Miller and Mukerji, 2003). MDA is a framework for software development aligned with MDD. The main characteristics of the framework are the definition of models as first class elements for the design and implementation of systems, and the definition of mappings between those models to allow transformations to be automated. To help manage the journey from 'real world' to code, the MDA architectural approach partitions models into those that are: (a) Computation Independent Models (CIMs), which represent the system domain, the business process, etc.; (b) Platform Independent Models (PIMs), which represent the system functionality but without taking into account any specific platform; and (c) Platform-Specific Models (PSMs), which represent the specifications described in the PIMs for some specific platform and technology. As per the ideal, the MDA approach proposes that code be automatically generated from the PSMs.

One of the main objectives of the MDA is to automate as much of possible the code generation as possible from PIMs and even from CIMs – this is possible if we are able to automate models and mappings. In this sense, recent history shows an increase in research activity. As a result, it is now possible to differentiate and choose between different approaches to automating mappings such as graph theory based transformations, mathematical models based transformations, etc. The main problem is that all of these approaches are *ad hoc* transformations for some specific platform – so maintaining the transformation process forces a re-implementation of the mappings. To solve this problem OMG is working in a new standard for model transformation called QVT (Query View Transformation) (OMG, 2005).

Benefits

The MDD approach can help to solve some traditional problems of Information Systems development. The main advantages are, perhaps, the facility for code generation and migration. These two benefits arise directly from the independence between models of different layers in the MDA. Since the system is described in the PIM layer independently of the platform, it is easily transportable from one platform to another. The degree of portability will depend, of course, of the existent tools for mappings to the chosen platform.

MDD could also help to solve interoperability problems. As is well known, different PSMs obtained from the same PIM could have relationships, called *bridges* in MDA, which it is necessary to preserve. With this aim, MDA proposes generating both the PSMs for the required platforms and also the bridges between them.

The research challenges

Such frameworks are not easily translated into practice however, and many research challenges with regard to MDD exist. For example, the computation independent layer is not well understood but implicitly makes strong demands in relation to understanding the types of things and relations that exist in the real world. Across the board, models also need to be developed at an appropriate level of (a) abstraction, (b) generalization and (c) precision and accuracy. Similarly, the transformations between models should be considered as first class models in their own right and demand strong semantic treatment. Another open issue is related to the transformations between CIMs and PIMs that could not be automatically performed because they need, in addition to the strong semantic treatment, the intervention of the designer.

In practical terms, the MDD approach therefore requires significant thinking in relation to both model development and the modelling process if it is to be of value to organizations. In this regard, we divide the open problems in three categories:

(a) Technological problems related with the basic tools needed to build information systems with an MDD approach. Among these problems are modelling languages, model transformation languages and tools supporting modelling and transformation between models.
(b) Engineering problems related to how to build information systems using MDD technologies – that is, using models and model transformations. Among the problems here is the process of modelling, quality of models, model management, etc.
(c) Experience problems related to the applicability of MDD to different domains reflecting learning from the experience.

Special issue

This special issue broaches recent challenges and advances in MDD Engineering and Technology as well as the application of the Model-Driven Techniques to different domains, such as Data Warehouses or Web Engineering. The special issue also broaches orthogonal aspects such as security and the quality in the MDD process. A total of 35 papers were submitted to the special issue, of which seven papers were selected following the review process (a 20% acceptance rate). Work comes from a variety of countries including Belgium, France, Spain, the United Kingdom and the United States. It is interesting to note that none of the papers selected addressed MDSD from a 'soup-to-nuts' MDA perspective – that is, developing an application from inception through to final code-generation. Our view on this observation is that it provides an indication that the state-of-the-art is still immature. Researchers are starting to experiment with aspects of the model-driven approach in 'safe' environments, but the approach is not robust enough for more ambitious projects. To that end, we intend that the special issue provides insight into the research challenges that lie ahead.

In very broad terms, the special issue is divided into themes where the papers deal with model transformation issues, applications of particular aspects of the model-driven approach and, as a variation, application in the context of the Web. We start the special issue with an examination of model differentiation techniques, which seek to identify differences and deal with mapping between models. There, Lin *et al*. from the *ATLAS group at INRIA, LINA and University of Nantes* (France), examine domain specific languages as an alternative to UML (which conforms to a single metamodel). Their work presents algorithms and tools to detect and potentially deal with differences in the metamodels that underlie different domain specific languages, providing facilities for graphical visualization of the detected differences. The second paper on this theme deals with platform transformation. There, Wagelaar and van der Straeten from *Vrije University* (Brussels), propose an ontological model that can be used to reason about platform dependencies in model transformations (in order that platform models can be reused) and how platform models can be integrated into a model-driven configuration framework.

In relation to the second theme, the third paper examines security and confidentiality in relation to data warehouses and online analytic processing. There, Fernandez-Medina *et al*. from the *University of Castilla-La Mancha* and *University of Alicante* (Spain), present an Access Control and Audit (ACA) model appropriate to conceptual multidimensional modelling (equivalent to the PIM level) in order to automate the enforcement of security considerations in model-driven systems. They extend the UML language to represent at PIM level for this ACA model, representing constraints by using the OSCL (Object Security Constrains Language). The fourth paper addresses the direct transformation of business models into running systems in an agent-based environment. There, Xiao and Greer, from the *University of Southampton* (U.K.) and *Queen's University Belfast* (U.K.) respectively, demonstrate that, when business changes occur, it should not be necessary to directly edit the low level code models (as code).

In relation to the last theme, the fifth paper presents a model-driven process underlying a Web engineering method. There Montero *et al*. from the *DEI Laboratory at Carlos III University of Madrid* (Spain) present ontologically based specifications that provide semantics and reasoning for both model-transformations and model validation – instantiated through a method and toolset. The sixth paper concentrates on navigability in Web applications. There Cachero *et al*., representing joint work between research groups at the *University of Alicante* (Spain), the *University of Castilla-La Mancha* (Spain) and the *University of Gent* (Belgium), present a model-driven approach to improve the reuse and adoption of navigability measures alongside a generic navigability model. Last, Moreno *et al*. from *Malaga University* (Spain), address the problem of integrating applications in a Web context. The authors propose a model-driven approach to deal with the adaptation between a Web application and external assets – in essence using design patterns to promote generic reuse.

Acknowledgements

We would like to thank all the reviewers that have contributed to the quality of this special issue. In part, our work on the special issue has been made possible through the framework of the GOLD project, financed by the Spanish Ministry of Science and Technology (TIN 2005-00010).

References

Bézivin J (2005) On the unification power of models. *Software and System Modeling (SoSym)* **4(2),** 171–188.

Kleppe A, Warmer J and Bast W (2003) *MDA Explained, the Model Driven Architecture: Practice and Promise*. Addison Wesley, Reading, MA.

Miller J and Mukerji J (eds) (2003) MDA Guide Version 1.0.1, Object Management Group, Framingham, Massachusetts, June 2003.

OMG (2005) 2nd revised submission to the MOF 2.0 QVT RFP, March 2005, OMG Document -ad/05-03-02, http://www.omg.org/cgi-bin/doc?ad/05-03-0.

Selic B (2003) The pragmatics of model-driven development. *IEEE Software* **20(5),** 19–25.