Evolution in Model-Driven Engineering

Louis M. Rose

May 14, 2009

Abstract

To be completed.

This report comprises XX words, as counted by detex | wc -w.

Contents

1 Introduction

Software evolution is a development activity in which a software system is updated in response to some external pressure (such as changing requirements). [Brooks 1986] observes that engineering increasingly complicated systems with traditional development approaches presents many challenges including a resistance to evolution. Software evolution is discussed in Section 1.1.

Model-Driven Architecture (MDA) is a software engineering framework defined by the Object Management Group (OMG) [OMG 2008]. MDA provides a set of standards for developing computer systems in a model-centric, or *model-driven*, fashion. In *model-driven engineering* (MDE), models are utilised as the primary software development artefact. Several approaches to MDE are prevalent today, such as [Stahl *et al.* 2006], [Kelly & Tolvanen 2008] and [Greenfield *et al.* 2004]. The approaches vary in the extent to which they follow the standards set out by MDA.

In MDE, the primary software development artefacts are *models*. A model is a description of a phenomenon of interest [Jackson 1996], and may have either a textual or graphical representation. A model provides an abstraction over an object, which enables engineers of differing disciplines to reason about that object [Kolovos *et al.* 2006]. Employing the MDA guidelines has been shown to significantly improve developer productivity and the correctness of software [Watson 2008].

1.1 Software Evolution in Model-Driven Engineering

In the past, studies [Erlikh 2000, Moad 1990] have found that the evolution of software can account for as much as 90% of a development budget; there is no reason to believe that the situation is different now. [Sjøberg 1993] identifies reasons for software evolution, which include addressing changing requirements, adapting to new technologies, and architectural restructuring.

Software development often involves constructing a system by combining numerous types of artefact (such as source and object code, build scripts, documentation and configuration settings). Artefacts depend on each other. Some examples of these dependencies from traditional development include: compiling object code from source code, generating documentation from source code, and deploying object code using a build script. When one artefact is changed, the development team updates the other artefacts accordingly. Here, this activity is termed *migration*.

MDE introduces additional challenges for controlling and managing evolution [**?**]. For example, MDE prescribes automated transformation from models to code. Transformation may be partial or complete; and may take a model direct to code or use intermediate models [Kleppe *et al.* 2003]. Any code or intermediate models generated by these transformations are interdependent with other development artefacts – e.g. a change to a model may have an impact on other models [Deursen *et al.* 2007]. The process of maintaining consistency between interdependent models is termed *model synchronisation*.

To specify a transformation for a model, the model must have a well-defined set of structural elements and rules that it must obey. This information can be specified in a structured artefact termed a *metamodel*. A model is dependent on its metamodel. When a metamodel is changed, its models may require

migration. The process of maintaining consistency between a model and its metamodel is termed *(model and metamodel) co-evolution*.

1.2 Research Aim

In traditional software development, some migration activities can be automated (e.g. background incremental compilation of source code to object code), while some must be performed manually (e.g. updating design documents after adding a new feature). MDA seeks to reduce the amount of manual migration required to develop software, but presently no tools for MDE fully support automated evolution; managing evolution in the context of MDE remains an open research problem.

The aim of my research is to develop structures and processes for managing evolutionary changes in the context of Model-Driven Engineering. In my progress report, I presented and discussed data from existing MDE projects. The analysis of this data led me to conclude that little data was available for studying model synchronisation. Consequently, I decided to focus on developing structures and processes for managing (model and metamodel) co-evolutionary changes.

2 Proposed Thesis Structure

2.1 Introduction

2.1.1 Motivation

2.1.2 Definitions

2.1.3 Research Aims

2.1.4 Research Method

2.2 Background

2.2.1 Metamodelling

2.2.2 Model-Driven Engineering

Automated relationships between models.

2.2.3 Model and Metamodel Consistency

2.3 Literature Review

2.3.1 Conclusion

High-level objectives and summary of methodology

2.4 Analysis

2.4.1 Locating Data

2.4.2 Existing Approaches

Experiments

2.4.3 Requirements Identification

2.5 Implementation

2.5.1 Deferred Model to Metamodel Binding

2.5.2 Metamodel-Independent Syntax (HUTN)

2.5.3 DSL for Automated Co-evolution

2.6 Evaluation

2.6.1 Case Study

2.6.2 Publications

2.6.3 Delivery through Eclipse

2.7 Conclusion

High-level. Summary of research objectives.

2.7.1 Achievement

2.7.2 Future work

3 Progress

3.1 Achievements

3.2 Plan

References

[Brooks 1986] Frederick P. Brooks. No silver bullet – essence and accident in software engineering (invited paper). In *Proc. International Federation for Information Processing*, pages 1069–1076, 1986.

[Deursen *et al.* 2007] Arie van Deursen, Eelco Visser, and Jos Warmer. Model-driven software evolution: A research agenda. In *Proc. Workshop on Model-Driven Software Evolution*, pages 41–49, 2007.

[Erlikh 2000] Len Erlikh. Leveraging legacy system dollars for e-business. *IT Professional*, 2(3):17–23, 2000.

[Greenfield *et al.* 2004] Jack Greenfield, Keith Short, Steve Cook, and Stuart Kent. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley, 2004.

[Jackson 1996] Michael Jackson. *Software Requirements and Specifications*. Addison-Wesley and ACM Press, 1996.

[Kelly & Tolvanen 2008] Steven Kelly and Juha-Pekka Tolvanen. *Domain-Specific Modelling*. Wiley, 2008.

[Kleppe *et al.* 2003] Anneke G. Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.

[Kolovos *et al.* 2006] Dimitrios S. Kolovos, Richard F. Paige, and Fiona A.C. Polack. The Epsilon Object Language (EOL). In *Proc. European Conference on Model Driven Architecture – Foundations and Application*, volume 4066 of *LNCS*, pages 128–142. Springer, 2006.

[Moad 1990] J Moad. Maintaining the competitive edge. *Datamation*, 36(4):61–66, 1990.

[OMG 2008] OMG. Object Management Group home page [online]. [Accessed 30 June 2008] Available at: http://www.omg.org, 2008.

[Sjøberg 1993] Dag I.K. Sjøberg. Quantifying schema evolution. *Information & Software Technology*, 35(1):35–44, 1993.

[Stahl *et al.* 2006] Thomas Stahl, Markus Voelter, and Krzysztof Czarnecki. *Model-Driven Software Development: Technology, Engineering, Management*. Wiley, 2006.

[Watson 2008] Andrew Watson. A brief history of MDA. *Upgrade*, 9(2), 2008.