Chapter 17
Matilda
A Generic and Customizable Framework for Direct Model Execution in Model-Driven Software Development

Hiroshi Wada
University of Massachusetts, USA

Junichi Suzuki
University of Massachusetts, USA

Adam Malinowski
Harvard University, USA

Katsuya Oba
OGIS International, Inc., USA

ABSTRACT

Traditional Model Driven Development (MDD) frameworks have three critical issues: (1) abstraction gap between modeling and programming layers, (2) a lack of traceability between models and programs, and (3) a lack of customizability to support various combinations of modeling technologies and implementation/deployment technologies. In order to address these issues, this chapter proposes a new MDD framework, called Matilda, which is a framework to build execution runtime engines (or virtual machines) for software models. It directly executes models defined with certain modeling technologies such as UML and BPMN by automatically transforming them to executable code. Matilda is designed based on the Pipes and Filters architectural pattern, which allows for configuring its structure and behavior flexibly by replacing one plugin with another one or changing the order of plugins. Also, plugins can be deployed on multiple network hosts and seamlessly connect them to form a pipeline. This facilitates distributed software development in which developers collaboratively work at physically dispersed places. This chapter overviews Matilda's architectural design, describes the implementations of Matilda-based virtual machines, and evaluates their performance.

DOI: 10.4018/978-1-60566-731-7.ch017

INTRODUCTION

Software modeling has advanced to the point where it can offer significant leverage to manage complexity and improve productivity in software development. A driving force in this advance is a series of mature modeling technologies. For example, the Unified Modeling Language (UML) provides a wide range of modeling notations and semantics used in various types of applications (UML Super Structure Specification 2.1.2, 2007). The Business Process Modeling Notation (BPMN) provides a set of well-defined notations and semantics for business process modeling (Business Process Modeling Notation (BPMN) 1.0, 2004). UML and BPMN allow developers to specify and communicate their application designs at a high level of abstraction. Using these modeling technologies, the notion of model-driven development (MDD) aims to graphically build application design models and transform them into running applications.

A key process in MDD is automated (or semi-automated) transformation of implementation independent models to lower-level models (or application code) specific to particular implementation/deployment technologies such as programming languages, databases, middleware and business process engines (Booch, Brown, Iyengar, Rumbaugh, & Selic, 2004; Sendall & Kozaczynki, 2003). Traditional MDD frameworks allow developers to model their applications with modeling languages such as UML and BPMN, generate skeleton code in a programming language such as Java, and manually complete the generated skeleton code by, for example, adding method code (Figure 1). There exist three critical research issues in traditional MDD frameworks: (1) abstraction gap between modeling and programming layers, (2) a lack of traceability between models and programs, and (3) a lack of customizability to support various combinations of modeling technologies and implementation technologies.

The first issue is that, when programmers complete generated skeleton code to the final (compilable) code, they often suffer from abstraction gap between modeling and programming layers because the granularity of skeleton code is usually much finer than that of models. Skeleton code tends to be complicated to read and maintain. Thus, it is hard for programmers to obtain a broad view of an application design, and they have to repeatedly

Figure 1. Traditional MDD process