Chapter II

MDA with xUML: Model Construction and Process Management

Boris Roussev
University of the Virgin Islands, USA

Abstract

xUML epitomizes the convergence of visual modeling with model manipulation programming. The results of this merger are executable models and model-driven software development. This chapter presents the fundamental notions of constructing executable domain models with xUML, as well as the principles of the MDA approach. In particular, we define the new roles of the developers in development processes based on MDA and the MDA activity workflow. We discuss also the output artifacts from each activity. As new technologies require new software development processes, we present an iterative and incremental model-driven process, combined with techniques for project planning and progress estimation based on BERT and ERNIE. We show how model executability creates congenial conditions for the application of higher-order cognitive skills in the software development process, and for the substitution of liberal creativity with design automation.
Introduction

C.A. Petri was the first to define formally the notion of communicating state machines in his PhD thesis, “Kommunikation mit Automaten,” back in 1962. His visual modeling language for modeling concurrency and synchronization, later known as Petri nets, is an extension of the Finite State Machine (FSM) theory. Even though Petri nets is an intuitive and powerful process coordination language, several pieces were crucially missing to bridge the Petri nets paradise to the real world. Petri nets lacked an information structure model such as class diagrams, a development process for effective product development from informal requirements such as Agile (Beck, 1999), and action semantics (other than transition firing rules). Action semantics defines a virtual machine for model interpretation, giving substance to the notion of model “executability.”

The Shlaer-Mellor method (Shlaer & Mellor, 1988, 1992), one of the first object-oriented (OO) analysis methods, uses class diagrams to represent the information structure of a system. This information model was influenced by the relational theory of data (Codd, 1970) and database modeling with entity-relationship diagrams (Chen, 1977). Shlaer and Mellor reinvented the idea of communicating FSMs in the context of OO by employing FSMs to abstract lifecycles of objects, whose progression is driven by external or internal asynchronous signals. They describe objects’ execution behavior as state procedures consisting of actions. The actions perform tasks on modeling elements; for example, they traverse an association link to retrieve the value of an attribute in a related instance, or generate signals to the FSMs of related objects. Shlaer and Mellor advanced the idea of composing complete systems out of executable models.

Shlaer and Mellor put at the top of the OO agenda the notion of model executability. Their method evolved into a pure object-oriented notation (Mellor & Balcer, 2002; Mellor et al., 2004), which is currently shaping up the future of UML.

Model-Driven Architecture (MDA) (Mellor & Balcer, 2002) is a term defined by OMG. The MDA approach to software development relies on Executable UML (xUML) (MDA, 2004), a UML profile with executable semantics. MDA distinguishes between Platform Independent Models (PIM) and Platform Specific Models (PSM).

In MDA, software does not need to be programmed at all, or at least not by humans. This can be achieved through the integration of a programming