Chapter 13
Foundations for MDA Case Tools

Liliana Favre
Universidad Nacional del Centro de la Pcia. de Buenos Aires, Argentina

Claudia Teresa Pereira
Universidad Nacional del Centro de la Pcia. de Buenos Aires, Argentina

Liliana Inés Martinez
Universidad Nacional del Centro de la Pcia. de Buenos Aires, Argentina

INTRODUCTION

The model driven architecture (MDA) is an initiative proposed by the object management group (OMG), which is emerging as a technical framework to improve productivity, portability, interoperability, and maintenance (MDA, 2003).

MDA promotes the use of models and model-to-model transformations for developing software systems. All artifacts, such as requirement specifications, architecture descriptions, design descriptions, and code are regarded as models. MDA distinguishes four main kinds of models: computation independent model (CIM), platform independent model (PIM), platform specific models (PSM), and implementation specific model (ISM).

A CIM describes a system from the computation independent viewpoint that focuses on the environment of and the requirements for the system. In general, it is called domain model. A PIM is a model that contains no reference to the platforms that are used to realize it. A PSM describes a system with full knowledge of the final implementation platform. In this context, a platform is “a set of subsystems and technologies that provide a coherent set of functionality which any application supported by that platform can use without concern for the details of how the functionality is implemented” (MDA, 2003, p. 2-3). PIMs and PSMs are expressed using the unified modeling language (UML) combined with the object constraint language (OCL) (Favre, 2003; OCL, 2004; UML, 2004).

The idea behind MDA is to manage the evolution from CIMs to PIMs and PSMs that can be used to generate executable components and applications. In MDA is crucial to define, manage,
Foundations for MDA Case Tools

and maintain traces and relationships between different models and automatically transform them and produce code that is complete and executable.

Metamodeling has become an essential technique in model-centric software development. The metamodeling framework for the UML itself is based on architecture with four layers: meta-metamodel, metamodel, model, and user objects. A metamodel is an explicit model of the constructs and rules needed to build specific models, its instances. A meta-metamodel defines a language to write metamodels. OCL can be used to attach consistency rules to models and metamodels. Related OMG standard metamodels and meta-metamodels such as meta object facility (MOF), software process engineering metamodel (SPEM) and common warehouse model (CWM) share a common design philosophy (CWM, 2001; MOF, 2005; SPEM, 2005).

MOF defines a common way for capturing all the diversity of modeling standards and interchange constructs. MOF uses an object modeling framework that is essentially a subset of the UML core. The four main modeling concepts are “classes, which model MOF metaobjects; associations, which model binary relationships between metaobjects; data types, which model other data; and packages, which modularize the models” (MOF, 2005, p. 2-6). The query, view, transformation (QVT) standard depends on MOF and OCL for specifying queries, views, and transformations. A query selects specific elements of a model, a view is a model derived from other model, and a model transformation is a specification of a mechanism to convert the elements of a model, into elements of another model, which can be instances of the same or different metamodels (QVT, 2003).

The success of MDA depends on the existence of CASE (computer-aided software engineering) tools that make a significant impact on software processes such as forward engineering and reverse engineering processes (CASE, 2006). This article explains the most important challenges to automate the processes that should be supported by MDA tools. We propose an integration of knowledge developed by the community of formal methods with MDA. We describe a rigorous framework that comprises the metamodeling notation NEREUS and bridges between MOF-metamodels and NEREUS, and between NEREUS and formal languages. NEREUS can be viewed as an intermediate notation open to many other formal specifications. We analyze metamodeling techniques for expressing model transformations such as refinements and refactorings. Our approach focuses on interoperability of formal languages in model driven development (MDD).

This article is organized as follow. We first analyze the limitations of the existing MDA-based CASE tools. Then, we describe the bases of rigorous MDA-based processes. Next, we show how the formalization of MOF metamodels and metamodel-based model transformations allows us automatic software generation. Finally, we highlight the key directions in which MDA is moving forward.

BACKGROUND

To date, there are about 120 UML CASE tools that vary widely in functionality, usability, performance, and platforms (CASE, 2006). Some of them can only help with the mechanics of drawing and exporting UML diagrams. The mainstream object-oriented CASE tools support forward engineering and reverse engineering processes and can help with the analysis of consistency between diagrams. Only a few UML tools include extension for real time modeling. The tool market around MDA tools is still in flux and only about 10% of them provide some support for MDA. Table 1 exemplifies a taxonomy of the UML CASE tools (CASE, 2006).

The current techniques available in the commercial tools do not allow generating complete
Related Content

**Particle Swarm Optimizer for High-Dimensional Data Clustering**
www.igi-global.com/chapter/particle-swarm-optimizer-high-dimensional/46390?camid=4v1a

**Self-Repair by Program Reconfiguration in VLIW Processor Architectures**
www.igi-global.com/chapter/self-repair-program-reconfiguration-vliw/51404?camid=4v1a

**Experiences in Software Engineering Education: Using Scrum, Agile Coaching, and Virtual Reality**
www.igi-global.com/chapter/experiences-in-software-engineering-education/192922?camid=4v1a

**SUMMING UP THE PARTS**
www.igi-global.com/chapter/summing-parts/49185?camid=4v1a