Chapter 10
Model-Driven Engineering, Services and Interactive Real-Time Applications

Luis Costa
SINTEF ICT, Norway

Neil Loughran
SINTEF ICT, Norway

Roy Grønmo
SINTEF ICT, Norway

ABSTRACT

Model-driven software engineering (MDE) has the basic assumption that the development of software systems from high-level abstractions along with the generation of low-level implementation code can improve the quality of the systems and at the same time reduce costs and improve time to market. This chapter provides an overview of MDE, state of the art approaches, standards, resources, and tools that support different aspects of model-driven software engineering: language development, modeling services, and real-time applications. The chapter concludes with a reflection over the main challenges faced by projects using the current MDE technologies, pointing out some promising directions for future developments.

MODEL DRIVEN ENGINEERING

Overview

Model-driven software engineering (MDE) (Kent, 2002) refers to the generation of software systems from high-level abstractions using *meta-models* and *models*. A meta-model provides the language abstraction of a system, while models, i.e. instances of the meta-models, allow the specification of an intended application based upon the constraints imposed by the meta-model. Typically, MDE is used in the building of a *domain specific language* (DSL) (Fowler & Pearson, 2010). A DSL, as opposed to a general purpose low-level language like Java or C++, is a high-level language containing abstractions that are specific to a certain application area or task. For example, the language

DOI: 10.4018/978-1-4666-4301-7.ch010

constructs for a DSL in the automotive domain may refer to concepts relating directly to automotive concepts (e.g. vehicle type, engine capacity, transmission, etc.).

The architecture of such languages typically consists of two key elements: the *abstract* and the *concrete* syntax. The abstract syntax is usually defined using a model. As this model is actually a model of other models it is called a meta-model. The meta-model describes the different concepts of the language and the relationships between them. The concrete syntax describes what syntactic elements constitute the language, or how the elements of the abstract syntax are to be presented or specified. One example is that in the Unified Modeling Language (UML) (OMG, 2010c) a Class is represented by a box. A language may have several concrete syntaxes for the same abstract syntax. For example it can have both a graphical notation (diagrams) and a textual notation. The field of language engineering is not new, but in recent years tool support has improved considerably. This eases the creation of both abstract and concrete syntaxes and the connections between them. Examples of such tools are the Eclipse Graphical Modeling Framework (GMF) (Eclipse, 2010b) and MetaEdit from MetaCase (Metacase, 2010). The Object Management Group (OMG) (OMG, 2010a), the standardisation organisation behind the UML, typically when standardising such specific languages requires a meta-model and a UML profile representing the concrete syntax.

In this chapter we describe a number of the state of the art approaches in modeling with respect to language development, services and real-time modeling.

MODELING TECHNOLOGIES

In this section we investigate a range of the state of the art in model-driven engineering techniques within the UML and Eclipse worlds.

MOF

The Meta Object Facilities (MOF) (OMG, 2010b) proposed by the OMG provides a *meta-metamodel* at its top layer. If we consider a meta-model as concepts for describing models, then a meta-metamodel can be considered as the concepts used for describing a given meta-model. MOF is based on the object-oriented paradigm and there are different flavours in existence: Complete MOF (CMOF), Essential MOF (EMOF) and Semantic MOF (SMOF). CMOF is the whole of the MOF specification, whereas the EMOF contains only a subset of the main elements of MOF. SMOF is currently a proposal for adding additional semantics to MOF. MOF is currently utilised by UML and Ecore, i.e. the meta-model of the Eclipse Modeling Framework (Eclipse, 2010a).

UML

The Unified Modeling Language (UML) (OMG, 2010c) is a standardized general-purpose modeling language used in software engineering to model systems. UML provides a standard notation to deal at a high level with several facets of complexity of a system.

The Language

A UML model consists of elements such as packages, classes, and instances. UML diagrams are graphical representations of parts of the UML model. UML diagrams are formed by graphical elements (nodes connected by paths) that represent elements in the UML model. In this sense, the different diagram types of UML (see Figure 1) try to define and express the functionality of a system from different points of view.

UML is defined using MOF and is a central aspect of MDE today as it is the most wide-spread modeling language in general, and it is commonly used to define models used for further processing through transformations and refinements.