Chapter 3
MDA, Metamodelling and Transformation

INTRODUCTION

MDA requires the ability to understand different languages such as general purpose languages, domain specific languages, modeling languages or programming languages. An underlying principle of MDA for integrating semantically in a unified and interoperable way such languages is using metamodelling techniques.

A metamodel describes a family of models whose elements are instances of the metaclasses of the respective metamodel. The kind of entities and relations defines the kind of metamodel, for instance:

- An ISM-Java metamodel includes entities (metaclasses) for classes, fields, operations, methods, constructors, parameters and interfaces. Methods and constructors are subtypes of operations. Interfaces are associated with classes.
- A PSM-Java metamodel distinguishes entities such as Java-metamodel entities and another entities such as associations.
- A RDBMS metamodel includes entities for schema, table, column, key and foreign key.

The OMG standard for defining models is the Meta-Object-Facility (MOF) metamodel (MOF, 2006). MOF is essential to define different modeling languages and metamodelling languages such as UML or MOF itself. It allows capturing all the diversity of modeling standards and interchange constructs that are used in MDA. A MOF-aware modeling tool can capture UML diagram elements in machine readable form allowing tools from multiple vendors to be used together on a single project.

DOI: 10.4018/978-1-61520-649-0.ch003
The initial diffusion of MDA was focused on its relation with UML as modeling language. However, there are UML users who do not use MDA, and MDA users who use other modeling languages such as Domain Specific Languages (DSL).

The essence of MDA is MOF that allows different kinds of software artifacts to be used together in a single project. It allows capturing all the diversity of modeling standards and interchange constructs that are used in MDA. MOF provides a metadata management framework, and a set of metadata services to enable the development and interoperability of models and metadata driven systems.

**MOF CONSTRUCTS**

The MOF modeling concepts are “classes, which model MOF meta-objects; associations, which model binary relations between meta-objects; Data Types, which model other data; and Packages, which modularize the models” (MOF, 2006, pp. 2-6). OCL can be used to attach consistency rules to metamodel components.

Next, we describe these constructs in detail.

**Classes**

Classes are type descriptions of “first class instance” MOF meta-objects. Instances of classes have object identity, state, and behavior. Classes can have three kinds of features which are attribute, operation and reference. They can also contain exceptions, constants, data types, constraints and other elements.

An attribute has properties such as type, name and multiplicity. Besides, it can contain flags such as “isChangeable” and “isDerived”. The first determines whether the client is provided with an explicit operation to set the attribute values and the latter, determines whether the contents of the notational value holder is derived from other state.

Operations are “hooks” for describing the class behavior. They simply specify the name, the type signatures by which the behavior is invoked. Operations have the following properties: name, a sequence of parameters including name, type and multiplicity, an optional return type and, a list of Exceptions that can be raised by an invocation. An attribute may be an optional-valued, single-valued, or multi-valued depending on its multiplicity specification. The multiplicity can also include the flags “is-Ordered” for indicating ordered attributes and “is-unique” for indicating whether instances with equal value are allowed in the given attribute or parameter.

Like UML, MOF provides class generalization. However, MOF imposes restrictions on generalization to ensure that it can be transformed into a range of implementation technologies:

- A class cannot generalize itself, either directly or indirectly
- A class cannot generalize another class if the subclass contains a model element with the same name as a model element contained or inherited by the superclass (i.e. no over-riding is allowed)
- When a class has multiple superclasses, no model elements contained or inherited by superclasses can have the same name.

MOF uses “abstract class” in the same sense as UML and other object-oriented programming languages.