Chapter 11
Summing Up the Parts

REVERSE ENGINEERING: A DIFFERENT POINT OF VIEW

This chapter summarizes the main results described in this book and challenges and strategic directions in MDA reverse engineering.

Reverse engineering is the process of analyzing software systems to extract software artifacts at a higher level of abstraction.

Nowadays, software and system engineering industry evolves to manage new platform technologies, design techniques and processes. Architectural framework for information integration and tool interoperability, such as MDA, had created the need to develop new analysis tools and specific techniques.

MDA is not itself a technology specification but it represents an evolving plan to achieve cohesive model-driven technology specifications. The original inspiration around the definition of MDA had to do with the middleware integration problem in internet. Beyond interoperability reasons, there are other good benefits to use MDA such as to improve the productivity, process quality and maintenance costs.

The outstanding ideas behind MDA are separating the specification of the system functionality from its implementation on specific platforms, managing the software evolution from abstract models to implementations increasing the degree of automation and achieving interoperability with multiple platforms, programming languages and formal languages.

DOI: 10.4018/978-1-61520-649-0.ch011
MDA distinguishes at least three main models: Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM) and Implementation Specific Model (ISM).

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 DSLs. The essence of MDA is MOF that allows different kinds of artifacts from multiple vendors to be used together in a same project.

MOF-metamodels are expressed as a combination of UML, OCL and natural language. MOF has no built-in semantics apart from the well-formedness rules in OCL and what can deduced from them. This form of specification does not make possible validating that specific metamodels, like UML metamodel, conform MOF. A combination of MOF metamodeling and formal specification can help us to address MDA reverse engineering process.

With the emergence of MDA, new approaches should be developed in order to reverse engineering, both platform independent and platform specific models, from object oriented code.

This book describes MDA reverse engineering processes based on the integration of traditional reverse engineering techniques, advanced metamodeling techniques and formal specification. A framework to reverse engineering MDA models from object oriented code that distinguishes three different levels of abstraction linked to models, metamodel and formal specification was proposed.

The model level includes code, PIMs and PSMs. A PIM is a model with a high level of abstraction that is independent of an implementation technology. A PSM is a tailored model to specify a system in terms of specific platform such J2EE or .NET. PIMs and PSMs are expressed in UML and OCL. The subset of UML diagrams that are useful for PSMs includes class diagram, object diagram, state diagram, interaction diagram and package diagram. On the other hand, a PIM can be expressed by means of use case diagrams, activity diagrams, interactions diagrams to model system processes and state diagrams to model lifecycle of the system entities. An ISM is a specification of the system in source code.

At model level, transformations are based on classical compiler construction techniques. They involve processes with different degrees of automation, which can go from totally automatic static analysis to human intervention requiring processes to dynamically analyze the resultant models. All the algorithms that deal with the reverse engineering share an analysis framework. The basic idea is to describe source code or models by an abstract language and perform a propagation analysis in a data-flow graph called, in this context, object-data flow. This static analysis is complemented with dynamic analysis supported by tracer tools.

The metamodel level includes MOF metamodels that describe the transformations at model level. MOF metamodels describe families of ISMs, PSMs and PIMs. Every ISM, PSM and PIM conforms to a MOF metamodel. Metamodel transformations are specified as OCL contracts between a source metamodel and a target metamodel. These contracts “control” the transformation consistency.

This book describes a formalization of MDA processes in terms of MOF-based metamodels and QVT-based transformations. Both, MOF and QVT, depends on UML metamodel, which in turn depends on OCL. That is to say, the formalization of MDA processes depends on various OMG standards. Some of them, such as OCL and QVT, involve imperative constructions that are hard to formalize. To avoid this inconvenient, we analyzed the graph of package dependencies to select a minimal set of packages that allows us to precisely define the semantics of MDA process in a way independent of imperative constructions. The formalization is based on a subset of OMG standard metamodels that we called MDA Infrastructure, including elements of UML Infrastructure, EssentialOCL, EMOF and the QVT-Core.