Chapter 10

MDA–Based Object–Oriented Reverse Engineering

INTRODUCTION

Reverse engineering is the process of analyzing software systems to extract software artifacts at a higher level of abstraction so that it is easier to understand them, e.g., for reengineering, modernizing, reuse, migration or documenting purposes.

This chapter describes an approach to reverse engineering object oriented code. A central idea in reverse engineering is exploiting the source code as the most reliable description both of the system behavior and of the organization and its business rules.

We propose an approach for MDA-based object oriented reverse engineering that integrates classical compiler techniques, metamodeling techniques and formal specification for recovering designs and architectures.

We analyze reverse engineering of PSMs and PIMs from object-oriented code. Models are expressed using UML and OCL. On the one hand, the subset of UML diagrams, that are useful for platform-dependent models, includes class diagram, object diagram, state diagram, interaction diagram (collaboration diagram and sequence diagram) and package diagram. On the other hand, a PIM can be expressed by means of use case diagrams, activity diagrams, interaction diagrams to model system processes and state diagrams to model lifecycle of the system entities.

DOI: 10.4018/978-1-61520-649-0.ch010
Reverse engineering involves 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. Then, we analyze static and dynamic analysis techniques for recovering models at different abstraction levels.

We show how MOF-based metamodels can be used to drive model recovery processes. Besides, considering that validation, verification and consistency analysis are crucial activities in the modernization of systems, we propose an algebraic formalization of these MOF-defined reverse engineering processes.

Next, we describe related work and the features of existing reverse engineering tools.

**RELATED WORK**

Many works had contributed to reverse engineering object oriented code. (Muller, Jahnke, Smith, Storey, Tilley, & Wong, 2000) presents a roadmap for reverse engineering research for the first decade of the 2000s. (Angyal, Lengyel, & Charaf, 2006) is an overview of the state-of-the-art of reverse engineering techniques. A more recent survey of existing work in the area of reverse engineering is (Canfora & Di Penta, 2007). This article compares existing work, discusses success and provides a road map for possible future developments in the area.

Fanta and Rajlich (1998) describe the reengineering of a deteriorated object-oriented industrial program written in C++. In order to deal with this problem, they designed and implemented several restructuring tools and used them in specific reengineering scenarios.

Systa (2000) describes an experimental environment to reverse engineer JAVA software that integrates dynamic and static information.

Demeyer, Ducasse, & Nierstrasz (2002) distinguish a variety of techniques for object-oriented reengineering based on patterns.

Qiao, Yang, Chu and Xu (2003) present an approach to bridging legacy systems to MDA that includes an architecture description language and a reverse engineering process.

Koehler, Hauser, Kapoor, Wu, and Kumaran (2003) describe a method that implements model driven transformations between particular platform-independent (business views) and platform-specific (IT architectural) models. On the PIM level, they use business process models and on the PSM level, the IT architectural models are service-oriented and focus on specific platform using Web service and workflows.

Gueheneuc (2004) proposes a study of class diagram constituents with respect to their recovery from object oriented code.

Boronat, Carsi and Ramos (2005) describe MOMENT, a rigorous framework for automatic legacy system migration in MDA.

MacDonald, Russell, and Atchison (2005) report on a project that assessed the feasibility of applying MDD to the evolution of a legacy system.

Deissenboeck and Ratiu (2006) show the first steps towards the definition of a metamodel that unifies a conceptual view on programs with the classical structure-based reverse engineering metamodels.

Tonella and Potrich (2005) provide a relevant overview of techniques that have been recently investigated and applied in the field of reverse engineering of object oriented code. They describe the algorithms involved in the recovery of UML diagrams from code and some of the techniques that can be adopted for their visualization. The algorithms deal with the reverse engineering of the following