Software Modernization and the State-of-the-Art and Challenges

S

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

Claudia Pereira
Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

Liliana Martinez
Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

INTRODUCTION

Software modernization, understood as technological and functional evolution of legacy systems, provides principles, methods, techniques and tools to support the transformation from an existing software system to a new one that satisfies new requirements.

Modernization is related to different processes such as migration, software refactoring, architecture restructuring, and mainly reverse engineering. Reverse engineering techniques allow supporting an integral part of the software modernization, specifically, the process of analyzing available software artifacts such as requirements, design, architectures, code or byte code, with the objective of extracting information and providing high-level views on the underlying system. Thus, software modernization starts from an existing implementation and requires an evaluation of every part of the system that could be transformed or implemented anew from scratch. Reverse engineering involves (re) discovering the functional, structural and behavioral semantics of a given artifact in order to document, maintain, improve or migrate them (Canfora & Di Penta, 2007).

Software modernization requires new technical frameworks for information integration and tool interoperation such as the Model Driven Development (MDD). MDD refers to a range of development approaches based on the use of software models as first-class entities. The most well-known realization of MDD is the OMG standard Model Driven Architecture (MDA) (MDA, 2013). 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 of model transformations and achieving interoperability with multiple platforms, programming languages and formal languages. The essence of MDA is the Meta Object Facility Metamodel (MOF) that allows different kinds of software artifacts to be used together in a single project (MOF, 2011). Models play a major role in MDA which distinguishes Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). Some authors also distinguish Implementation Specific Model (ISM) as a description (specification) of the system in source code. To express transformations, OMG defined the MOF 2.0 Query, View, Transformation (QVT) metamodel (QVT, 2011).

Software modernization can be summarized as follows. First, information is extracted out of the system artifacts. Second, this information is analyzed in order to take adequate modernization decisions and finally, the information must be transformed to new artifacts. These steps are supported by metamodels to describe existing systems, discoverers to automatically create models of these systems and, tools to understand and transform complex models created out of existing systems. OMG is involved in the definition of standards to successfully modernize existing information systems. In this direction, OMG Architecture-Driven Modernization Task Force (ADMTF) is developing a set of specifications and promoting industry consensus on modernization of existing applications (ADM, 2012).

DOI: 10.4018/978-1-4666-5888-2.ch723

This article analyzes ADM-based software modernization. It provides an overview of the-state-of-the-art in reverse engineering techniques and software modernization techniques. Taxonomy of different techniques is described. We describe how traditional techniques such as static and dynamic analysis can be integrated with ADM standards playing a central role in the evolution of software. Foundations of ADM CASE tools to develop industrial size software are analyzed. Finally, challenges and strategic directions in software modernization are included.

BACKGROUND

25 years ago, modernization focused mainly on reverse engineering for recovering high-level architectures or diagrams from procedural code to face up with problems such as comprehending data structures or databases or the Y2K problem.

Many works had contributed to reverse engineering object-oriented code. Muller, Jahnke, Smith, Storey, Tilley, and Wong (2000) present a roadmap for reverse engineering research for the first decade of the 2000s. Fanta and Rajlich (1998) describe the reengineering of a deteriorated object-oriented industrial program written in C++. Systa (2000) describes an experimental environment to reverse engineer JAVA software integrating dynamic and static information. Demeyer, Ducasse, and Nierstrasz (2002) distinguish a variety of techniques for object-oriented reengineering based on patterns.

When the Unified Modeling Language (UML) (UML, 2011) emerged, a new problem was how to extract higher-level views of the system expressed by different kind of UML diagrams. The diagrams that could be reverse-engineered in this way were partial. A new challenge was how to identify different relationships (e.g. dependency, association, aggregation and composition). Canfora and Di Penta (2007) present a relevant survey that compares existing work in the reverse engineering area, discuss success stories and main achievements, and provide a road map for possible future developments in this area. Fleurey, Breton, Baudry, Nicolas, and Jézéquel (2007) report on the use of MDE as an efficient, flexible and reliable approach for a software migration process. The described process, developed at Sodifrance, includes automatic analysis of the existing code, reverse engineering of abstract high-level models, model transformation to target platform models and code generation.

Other related work is specifically linked to ADM. For example, Cánovas Izquierdo and García Molina (2009) present a process to extract models that conform to ADM standards such as the Knowledge Discovery Metamodel (KDM) and the Abstract Syntax Tree Metamodel (ASTM) (KDM, 2012) (ASTM, 2011).

Barbier, Deltombe, Parisy, and Youbi (2011) describe a model driven reverse engineering method based on metamodeling and model transformation. The method was implemented in the BLU AGE® Reverse module, an Eclipse IDE plugin. In this approach, a textual DSL is constructed to later describe source code as formal KDM models. Next, these KDM models are transformed to UML PIMs. Authors generalize this method by extending KDM along with an implementation of the ASTM. Authors state that the link between KDM and ASTM is not clear even confusing.

Many CASE tools support reverse engineering, however, they only use more basic notational features with a direct code representation and produce very large diagrams (CASE MDA, 2012). The Eclipse-MDT MoDisco open source project is considered by ADMTF as the reference provider for implementations of several of its standards. It is a reusable and extensible model-based framework that facilitates the construction of reverse engineering applications (MoDisco, 2012).

Few MDA-based CASE tools support any of the QVT languages. As an example, IBM Rational Software Architect and Spark System Enterprise Architect do not implement QVT. Other tools partially support QVT, for instance Together allows defining and modifying transformations model-to-model (M2M) and model-to-text (M2T) that are QVT-Operational compliant. Medini QVT partially implements QVT (Medini, 2012). It is integrated with Eclipse and allows the execution of transformations expressed in the QVT-Relation language (CASE MDA, 2012).

The MMT (Model-to-Model Transformation) Eclipse project is a subproject of the top-level Eclipse Modeling Project that provides a framework for model-to-model transformation languages. Transformations are executed by transformation engines that are plugged into the Eclipse Modeling infrastructure. The main transformation engines developed in the scope of that project are ATL and QVT (ATL, 2012). ATL is a model transformation language and toolkit developed by ATLAS INRIA & LINA research group.