Chapter 7

Model-Driven Software Migration:
Process Model, Tool Support, and Application

Andreas Fuhr  
University of Koblenz-Landau, Germany

Tassilo Horn  
University of Koblenz-Landau, Germany

Andreas Winter  
Carl von Ossietzky University, Germany

Uwe Kaiser  
pro et con GmbH, Germany

Uwe Erdmenger  
pro et con GmbH, Germany

Volker Riediger  
University of Koblenz-Landau, Germany

Werner Teppe  
Amadeus Germany GmbH, Germany

ABSTRACT

Established software systems usually represent important assets, which are worth preserving in new software structures, to combine already proven functionality with the benefits of new technologies. The SOAMIG project is aimed at developing an adaptable migration process model with an accompanying tool support based on model-driven technologies. This process model, which combines reverse and forward engineering techniques, was applied in two different case studies on migrating a monolithic software system to service-oriented architecture and to a transformation-based language migration from COBOL to Java.

INTRODUCTION

Most commercially built information systems are based on traditional technologies preventing them from unfolding their full potential in future software development. Service-Oriented Architectures (SOA) (Arsanjani, et al., 2008; Gold, Knight, Mohan, & Munro, 2004) provide a modern and promising approach to increase flexibility in software adaptation, maintenance and evolution by referring to the underlying business processes to be supported by the software systems. Necessary functionality is specified by services which
are implemented by loosely coupled components that can easily be rearranged to fulfill changing user needs.

Current software systems, already established and time-proven, are not implemented in a service-oriented manner. These legacy systems usually provide software functionality by monolithic and deeply interwoven modules. Without significant reengineering, these systems will not benefit from Service-oriented Architectures. Referring to the staged model for the software life cycle (Rajlich & Bennett, 2000), appropriate techniques are requested to keep them in evolution instead of passing them to service or replacing them.

Software migration, i.e., transferring existing software systems to a new environment, provides elaborated methods and techniques allowing already established software systems to benefit from the advantages of new technologies. Migration does not change the functionality of the original systems; it only changes their embedding in appropriate environments. Although migration activities use the same technologies as reengineering activities, migration and reengineering have to be distinguished from each other. Whereas reengineering focuses on improving software quality, migration only focuses on conserving legacy systems in new environments (Sneed, Wolf, & Heilmann, 2010). Migrating legacy systems to SOA, as depicted in this section, allows the reuse of already established and proven software assets and their integration with newly created services, including their further evolution (Fuhr, Horn, Riediger, & Winter, 2011). Orchestrating migrated and newly realized service components to support new business processes is done in additional projects; the migration only enables the reuse of legacy functionality in a new service-oriented embedding. Migration to a new environment is not viewed as activity to improve code quality. This allows clearly separating migration, i.e., transformation, tasks from reengineering, i.e., quality improvement, tasks. Nevertheless, migration projects should be framed by reengineering projects to improve the legacy’s quality prior to (pre-renovation) and the migrated system’s quality after migration (post-renovation).

Migrating to SOA addresses architecture migration, which describes essentially reworking the software structure. Here, software assets have to be identified and abstracted to services supporting business processes. Furthermore, code has to be reorganized in new service components implementing the extracted services. These migrations deal with various aspects of software systems: data and databases, code and users and system interfaces are affected during migration. Occasionally, migration to SOA comes with language migration where—next to structural adaptations—programming languages like COBOL to Java are changed as well.

Migrating software systems requires a deliberated process taking into account the different aspects of migration projects. This process has to be supported by an adequate migration tool support. Empirical studies in the METAMORPHOS project (de Lucia & Tortora, 2010) impressively demonstrated the increase of productivity in migration projects by using tailored migration tool suites instead of traditional software development frameworks. SOAMIG’s objective was to define such a process and to present and evaluate a corresponding tool chain. Here, the main purpose was to show its applicability. Discussion, assessment, and improvement of migration and reengineering efforts were not part of the SOAMIG project. The improvement of maintenance efforts based on appropriate methodology and tooling is described elsewhere (see e.g. Borchers, 1997; de Lucia & Tortora, 2010).

Current process models in software development are primarily focused on the initial development of software systems. More modern, iterative and agile approaches like Unified Process (Kruchten, 2003), Extreme Programming (Beck & Andres, 2004) or SCRUM (Schwaber, 2004) view maintenance and migration activities only as additional iterations. These process models do not
Related Content

CONFU: Configuration Fuzzing Testing Framework for Software Vulnerability Detection
[www.igi-global.com/article/confu-configuration-fuzzing-testing-framework/46151?camid=4v1a](www.igi-global.com/article/confu-configuration-fuzzing-testing-framework/46151?camid=4v1a)

A Method for Detecting Bad Smells and its Application to Software Engineering Education
[www.igi-global.com/article/a-method-for-detecting-bad-smells-and-its-application-to-software-engineering-education/122789?camid=4v1a](www.igi-global.com/article/a-method-for-detecting-bad-smells-and-its-application-to-software-engineering-education/122789?camid=4v1a)

A Complete Routing Simulator for Digital Microfluidic Biochip
[www.igi-global.com/article/a-complete-routing-simulator-for-digital-microfluidic-biochip/231581?camid=4v1a](www.igi-global.com/article/a-complete-routing-simulator-for-digital-microfluidic-biochip/231581?camid=4v1a)

Building an LMS with Ubiquitous Software
[www.igi-global.com/chapter/building-lms-ubiquitous-software/29414?camid=4v1a](www.igi-global.com/chapter/building-lms-ubiquitous-software/29414?camid=4v1a)