Concept-Based Engineering of Situation-Specific Migration Methods

Marvin Grieger[1] Masud Fazal-Baqaie[2] Gregor Engels[3] Markus Klenke[4]

Abstract: Software migration methods enable to reuse legacy systems by transferring them into new environments. Thereby, the method used needs to fit to the project's situation by considering conceptual differences between the source and target environment and automating parts of the migration whenever suitable. Using an inappropriate migration method may lead to a decreased software quality or increased effort. Various method engineering approaches have been proposed to support the development of situation-specific migration methods. However, most do not provide a sufficient degree of flexibility when developing a method or fall short in guiding the endeavor. To address this problem, we introduce a situational method engineering framework to guide the development of model-driven migration methods by assembling predefined buildings blocks. The development is centered around the identification of concepts within a legacy system and the selection of suitable migration strategies. We evaluate the framework by an industrial project in which we migrated a legacy system from the domain of real estates to a new environment.

1 Overview

If an existing software system does not realize all of its requirements, but is still valuable to ongoing business, it has become legacy. This might be due to the fact that the underlying technology restricts the fulfillment of new requirements that arose over time. A proven solution is to migrate the existing system into a new environment. The migration is performed by establishing a migration project during which a *migration method* is enacted. The method specifies the activities to perform, roles to involve, tools to apply, and artifacts to generate in order to systematically transfer the legacy system into the new environment.

Using a migration method that fits to the project's situation is essential, as the method determines the efficiency and effectiveness of the overall migration project. To support the development of situation-specific methods, various method engineering approaches have been developed over time. However, we identified that existing approaches mainly suffer from two shortcomings [GFB15]: First, they do not provide a sufficient degree of flexibility when developing a method. Therefore, a fine-grained adaptation of the method for the situation at hand is often not possible. Second, they fall short in providing sufficient guidance on how to develop a method, making the endeavor error-prone.

[1] s-lab – Software Quality Lab, Paderborn University, Zukunftsmeile 1, 33102 Paderborn, Germany, grieger@s-lab.uni-paderborn.de
[2] S&N CQM Consulting & Services GmbH, Klingenderstr. 5, 33100 Paderborn, Germany, masud.fazal-baqaie@sn-cqm.de
[3] s-lab – Software Quality Lab, Paderborn University, Zukunftsmeile 1, 33102 Paderborn, Germany, engels@s-lab.uni-paderborn.de
[4] TEAM GmbH, Hermann-Löns-Straße 88, 33104 Paderborn, Germany, mke@team-pb.de

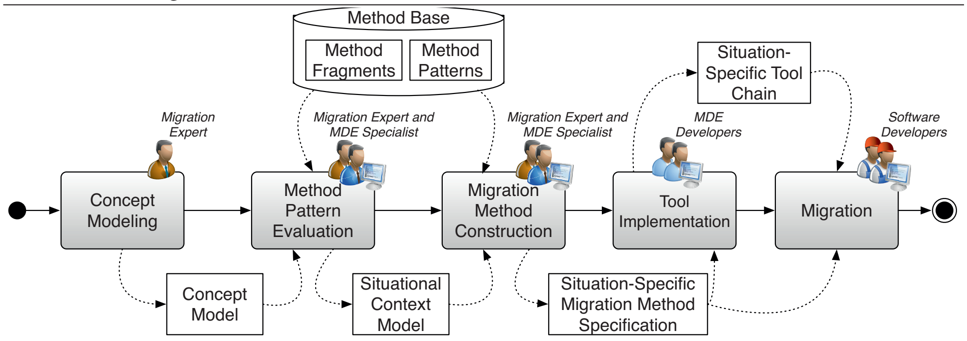
Method Base
Method Fragments
Method Patterns
Migration Expert
Migration Expert and MDE Specialist
Migration Expert and MDE Specialist
MDE Developers
Software Developers
Situation-Specific Tool Chain
Concept Modeling
Method Pattern Evaluation
Migration Method Construction
Tool Implementation
Migration
Concept Model
Situational Context Model
Situation-Specific Migration Method Specification

Fig. 1: Core activities of the method engineering framework

In this talk, a Situational Method Engineering (SME) framework that guides the development and enactment of situation-specific migration methods is introduced [Gr16]. An overview of the associated process is shown in Figure 1. The process begins with the activity called *Concept Modeling*. Thereby the concepts, i.e., the functionalities that are present within the system to migrate, are modeled. By focusing on the conceptual level we aim to abstract from the technology-specific realization. This enables to develop effective migration methods by choosing a suitable migration strategy. We propose a set of strategies that are encoded by *Method Patterns* and stored in the *Method Base*. Intuitively, the patterns represent construction guidelines for methods that follow an associated strategy. Before choosing a pattern for a concept, we aim to assess its implication during the *Method Pattern Evaluation* activity. This allows to make informed decisions in the subsequent *Migration Method Construction* activity. Thereby, a pattern is chosen for each concept and multiple patterns are integrated. If the resulting specification of the constructed method indicates that some parts of the migration are automated, then a corresponding model-driven tool chain is realized during the *Tool Implementation* activity. In the last activity called *Migration*, the developed method is enacted and the legacy system is transferred to the new environment. Thereby, developed tools get used and associated developers are included.

We evaluated the framework by applying it in an industrial context. In particular, we constructed and enacted a migration method to transform a legacy system from the domain of real estates. Due to the use of model-driven engineering and the focus on the conceptual level, we were able to automate parts of the migration while still ensuring a high quality of the resulting system.

References

[GFB15] Grieger, Marvin; Fazal-Baqaie, Masud: Towards a Framework for the Modular Construction of Situation-Specific Software Transformation Methods. 35(2):41–42, 2015. Proceedings of the 17th Workshop Software-Reengineering and Evolution (WSRE).

[Gr16] Grieger, Marvin; Fazal-Baqaie, Masud; Engels, Gregor; Klenke, Markus: Concept-Based Engineering of Situation-Specific Migration Methods. In: Proceedings of the 15th International Conference on Software Reuse (ICSR). volume 9679 of Lecture Notes in Computer Science. Springer, pp. 199–214, 2016.