Towards a Framework for the Modular Construction of Situation-Specific Software Transformation Methods

Marvin Grieger, Masud Fazal-Baqaie
Universität Paderborn, s-lab – Software Quality Lab
Zukunftsmeile 1, 33102 Paderborn
{mgrieger, mfazal-baqaie}@s-lab.upb.de

Abstract

Software transformation methods are enacted during a migration project to perform the technical transition of a legacy system to a new environment. A critical task of each project is to construct a situation-specific transformation method. In this paper, we categorize current Situational Method Engineering (SME) approaches that support the construction of situation-specific transformation methods according to their degree of controlled flexibility. Based on the findings, we introduce a method engineering framework that enables the modular construction of software transformation methods.

1 Introduction

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. As new development is risky and error-prone, a proven solution is to migrate the existing system into a new environment, which is performed in the context of a migration project. The technical transition of a system is achieved by enacting a *transformation method* which defines activities to perform, artifacts to create, tools to use, roles to involve and techniques to apply. As those methods are used to perform some kind of reengineering [1], they are instances of the well-established horseshoe-model [2].

Constructing a situation-specific transformation method when migrating a software system is critical, as it influences the efficiency and effectiveness of the overall migration project. Efficiency in this context relates to properties of the process to perform the transformation, e.g. the effort required, while effectiveness refers to properties of the migrated system, e.g. its software quality. Besides being a critical task, the construction of a transformation method is also a complex one. Consider for example that source and target environment differ in terms of the programming language. Performing a migration using a horseshoe-based process requires, among other things, to determine the abstraction level to use. A transformation on a syntactical level can be efficient, but ineffective. On the one hand it only requires to develop parsers, code generators and a mapping between the syntactic elements of the languages involved, and enables to transform large parts of the system automatically. On the other hand however, as the amount of information on a syntactical level is limited, it might not be possible to adapt the system to the new environment, e.g. in case of a migration from a monolithic to a layered architecture, making the method ineffective [3]. Using a higher level of abstraction can increase the effectiveness, e.g. by determining for each part of the system to which architectural layer it belongs. This will, however, reduce the efficiency of the method, as sophisticated program analysis techniques are required. Beside the abstraction level to use, a decision needs to be made on whether to automate the transformation at all. If an automatic transformation would be either inefficient or ineffective, a guided manual transformation is a possible alternative. In general, transformation methods are semi-automatic, using multiple abstraction levels jointly. Thus, constructing for a given project an efficient and effective, that is, situation-specific transformation method remains a critical but complex task in every migration project.

2 Situation-Specific Engineering of Transformation Methods

Situational Method Engineering (SME) approaches support the construction of situation-specific methods by providing reusable methods that have been successfully applied in practice, as well as guidance on how to adapt them to a new situation encountered. In [4] a categorization is introduced by which SME approaches are categorized based on the degree of *controlled flexibility* that describes the extent to which a situation-specific adaptation is possible. The adaptation needs to be *controlled* to ensure a high quality of the resulting method. Figure 1 illustrates this categorization applied on SME approaches that support the construction of software transformation methods.

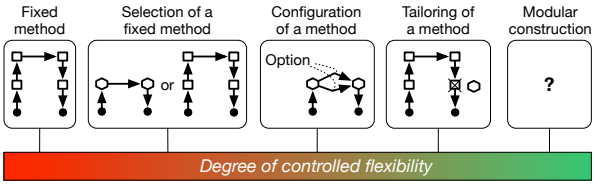

Figure 1: Categorization of method engineering approaches according to their degree of controlled-flexibility, based on [4]

The reuse of *fixed* methods can be seen as an edge case, SME approaches belonging to this category have the lowest degree of flexibility. *Fixed* methods are those who do not foresee any situation-specific adaptation as they describe a static set of activities to perform, tools to use, artifacts to generate, roles to include or techniques to apply in order to transform a legacy system. Such a method can either be described specifically, making it

only applicable to few situations, or generically, requiring situation-specific concretization. In any case the assumed situational context is often only described implicitly. If the situational context for a set of fixed methods is made explicit, it allows a *selection* of the most suitable one, which can be seen as a more flexible SME approach to perform situation-specific adaptation. But, since the resulting method will be fixed, the adaptability of approaches belonging to this category is still limited. In contrast, the definition of a base method that allows *configuration* or *tailoring* provides a higher degree of flexibility. While approaches of the former category aim at configuring foreseen variation points, approaches of the latter category can allow performing arbitrary, but well-defined change operations to the base method. This is achieved by providing a formal description and corresponding tools. However, if many changes to the base method are required, e.g. with novel content, or if no guidance is given on how to assure the quality of the resulting method, constructing a situation-specific method becomes complex and error-prone.

These drawbacks are addressed by SME approaches that enable the *modular construction* of transformation methods by assembling predefined method parts. In addition to the method parts, approaches of this category need to provide assembly guidelines and quality assurance capabilities for constructed methods. Unfortunately, approaches for the modular construction of transformation methods are hardly available. As transformation methods are a specific kind of reengineering methods, one could argue that software reengineering frameworks can be used to construct transformation methods in a modular manner. Although these frameworks are useful to implement tools that are part of transformation methods, they fall short in providing guidance on how to systematically construct the method itself and in assuring its quality.

3 Towards a Framework for the Modular Construction of Transformation Methods

We address this problem by developing a method engineering framework that enables the modular construction of transformation methods and thereby provides a high degree of controlled flexibility. An overview of the intended method engineering process is shown in figure 2.

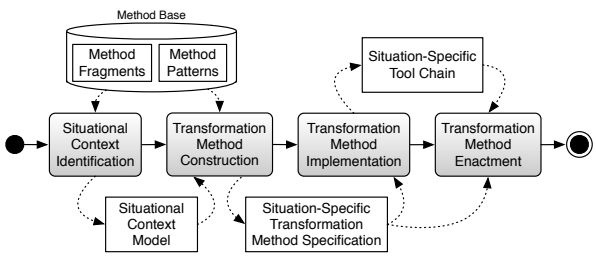

Figure 2: Method engineering process for the modular construction of situation-specific transformation methods

In the beginning, the situational context of the migration project needs to be identified, examples being characteristics of the legacy system, the change of the environment, or goals of stakeholders. The knowledge about the context is an essential prerequisite in order to construct a situation-specific transformation method. Using this knowledge, the transformation method is constructed before related tools are implemented. The construction is supported by a method base, which is a repository that contains predefined transformation method parts and transformation method patterns. While method parts are building blocks of a method of any granularity, method patterns describe methodological or quality aspects that shall be incorporated into the method [5]. For this purpose, a method pattern defines constraints over the method parts, e.g., by defining which ones to use. Adhering to these constraints during the construction allows to ensure properties of the resulting method. In addition, the patterns provide a guideline for the construction of a transformation method, since they are required to be selected, configured and integrated during the construction process. As a last step, the quality of the resulting method is validated. After the transformation method is constructed, related tools are implemented, i.e., the corresponding tool chain is initialized where necessary. As a last step, the transformation method is enacted to perform the transformation.

4 Conclusion and Future Work

In this paper, we characterized *Situational Method Engineering* (SME) approaches that support the construction of situation-specific transformation methods according to their degree of *controlled flexibility*. We concluded that current approaches have some shortcomings which we address by introducing a method engineering framework that enables the modular construction of transformation methods. We aim to develop and apply the framework in the MoSAiC project, which is supported by the Deutsche Forschungsgemeinschaft (DFG) under grants EB 119/11-1 and EN 184/6-1.

5 Literature

[1] E. J. Chikofsky and J. H. I. Cross, “Reverse engineering and design recovery: a taxonomy,” IEEE Software, vol. 7, no. 1, pp. 13–17, 1990

[2] R. Kazman, S. G. Woods, and S. J. Carrière, “Requirements for Integrating Software Architecture and Reengineering Models: CORUM II,” in Proc. of WCRE 1998, pp. 154– 163.

[3] F. Fleurey, E. Breton, B. Baudry, A. Nicolas, and J. Jézéquel, “Model-Driven Engineering for Software Migration in a Large Industrial Context,” in Proc. of MODELS 2007, pp. 482–497.

[4] Frank Harmsen, Sjaak Brinkkemper, and Han Oei. “Situational method engineering for information system project approaches.” In Proc. of CRIS 1994, pp. 169–194.

[5] Masud Fazal-Baqaie, Markus Luckey, and Gregor Engels. “Assembly-Based Method Engineering with Method Patterns.” In Proc. of SE 2013, pp. 435–444.