Architectural Restructuring by Semi-Automatic Clustering to Facilitate Migration towards a Service-oriented Architecture

Marvin Grieger, Stefan Sauer
Universität Paderborn, s-lab – Software Quality Lab
Zukunftsmeile 1, 33102 Paderborn
{mgrieger, sauer}@s-lab.upb.de

Markus Klenke
TEAM GmbH
Hermann-Löns-Straße 88, 33104 Paderborn
mke@team-pb.de

1 Introduction

Enterprises are nowadays increasingly faced with the fact that their information systems have become legacy. A common solution to this problem is to migrate the existing application to a new environment, e.g. to a new platform. Modern platforms often employ the architectural style of service-oriented architecture (SOA), in which an application is composed of loosely coupled entities that provide functionality as a service. The challenge in migrating towards a service-oriented architecture is to identify these entities in the legacy application and restructure its architecture accordingly.

In this paper we describe a semi-automatic process which combines hierarchical and partitioning clustering in order to improve an initial service design. The result is used to restructure the legacy application during the migration process. The purpose of the approach is to create a maintainable, service-oriented architecture. It is being developed in the context of an industrial project in which a model-driven software migration process and tool chain is built [1]. It will be applied on a set of enterprise legacy applications.

The paper is structured as follows: The case study is briefly introduced in Section 2. In Section 3 we describe our approach using an example. Section 4 discusses related work in this area. Thereafter we present preliminary results and outline future work in Section 5.

2 Case Study

The case study is based on a platform migration from Oracle Forms to Oracle ADF. On the source platform, the main concept of an application is to provide masks as user interfaces to interact with an underlying database. Applications consist of multiple modules, where each module is implemented separately. A module consists of module components, for which we assume that each includes exactly one mask as well as related data connections and business logic. In migrating towards the target platform, services have to be identified. As the user realizes business processes by traversing masks, a single module provides functionality that is related to one or more sequential activities in the underlying process. Thus, a module can be seen as a *business service*, according to the classification of service types given in [2].

Mapping modules to *services* results in applications that are hard to maintain, since many fine-grained services emerge. Therefore, *composite services* need to be identified, which aggregate and orchestrate the *business services*. In addition, since the capabilities of the legacy platform for software reuse were quite limited, code clones are omnipresent in every application we examined. Removing these clones by extracting the functionality and moving it into one single service will again increase the maintainability of the resulting application and additionally reduce the effort of manual reimplementation that is necessary.

3 Restructuring Process

The general idea of the approach is illustrated in Figure 1. As described in Section 2, there exist fine-grained *business services* as an initial service design. Additionally, we assume the existence of a dedicated *composite service*, which is externally callable as an entry point for an end user. The initial service design can be seen in Figure 1 (I).

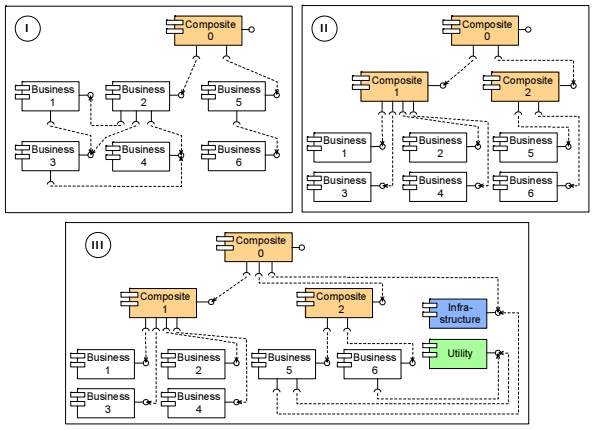

Figure 1: Illustrative example

The process related to our approach is shown in Figure 2. Based on the initial service design, hierarchical and partitioning clustering are performed to improve it gradually. These two activities consist of multiple steps which are described in detail subsequently.

3.1 Hierarchical Clustering

The first step of the restructuring process is to perform *hierarchical clustering* based on the dependencies between the services. As the dependencies constitute navigation flows, the resulting clusters are expected to ag-

gregate services that are related in terms of the underlying business process. The clusters are then implemented by introducing additional *composite services*, as can be seen in Figure 1 (II).

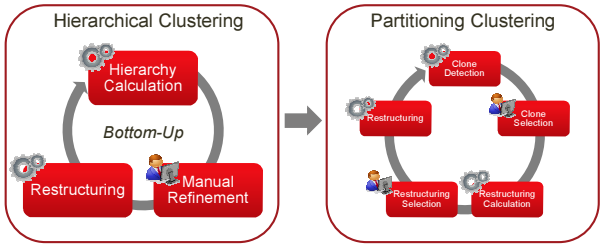

Figure 2: Restructuring process

As illustrated in Figure 2, hierarchical clustering consists of three activities, which are performed repeatedly. Each iteration starts by calculating new clusters for a hierarchy layer. This is performed automatically using a clustering algorithm. The result gets refined manually since it may not be optimal. This is due to the fact that clustering is performed on navigation flows rather than the business processes itself. The calculated clusters are then realized by introducing *composite services*. For that, predefined restructuring rules are executed. Depending on the size of the application, further iterations may be performed.

3.2 Partitioning Clustering

The second step of the restructuring process is to detect and remove software clones. This is achieved by partitioning the functionalities of multiple services into individual and common parts. Common parts are cut out and moved to a single service. This might be an existing one or a newly created *utility* or *infrastructure service*. The purpose of these types of services is to provide cross-cutting domain or technical functionalities that are used across different services. Therefore, these services do not belong to a specific layer calculated during the hierarchical clustering. The result of this step is shown in Figure 1 (III).

The *partitioning clustering* consists of five activities that are executed repeatedly, as shown in Figure 2. First, automatic clone detection is performed to suggest candidates of software clones. In order to do this, a strategy has to be defined where to search for clones, i.e., which services are being compared and for which functionalities clones are being detected. The reengineer thereafter selects clones that are to be removed. Based on this information, an algorithm determines restructuring possibilities, i.e., where to implement the functionality that should be cut out. As a result, it may either suggest the reengineer to move the functionality to an existing service or to create a new one. After the reengineer performs a decision, the restructuring is applied by executing a set of predefined restructuring rules.

4 Related Work

Migration towards SOA is an active area of research [3][4][5]. In contrast to most of the existing approaches, we assume that an initial service design exists, based on a mapping of existing legacy structures. As a result, we focus on identifying *composite services* instead of *business services*. In addition, we consider the use of clone detection for further refinement of the service design. To the best of our knowledge, no approach has combined these techniques in the context of a migration towards SOA.

5 Preliminary Results and Future Work

This approach is being developed in an industrial context. It has already been applied in the migration of an enterprise legacy system. Based on the described process, systematic restructuring steps have been derived which were manually applied on an application. We were able to show that the restructured service design was better in terms of increased maintainability and reduced effort for manual reimplementation, compared to a migration without restructuring.

The initial service design is improved by introducing new services and moving functionality between existing ones. In practice, a merge operation for small services is desired which may complement the approach. We are currently working on a tool that supports the activities described in the process. Automation will reduce the effort of restructuring and reduce the risk of manual errors. Afterwards, we will evaluate the approach with multiple applications.

6 Literature

[1] Grieger, M.; Güldali, B.; Sauer, S.: Sichern der Zukunftsfähigkeit bei der Migration von Legacy-Systemen durch modellgetriebene Softwareentwicklung. In Softwaretechnik-Trends, vol. 32, no. 2, pp. 37-38, 2012.

[2] Alahmari, S.; Zaluska, E.; De Roure, D.: A service identification framework for legacy system migration into SOA. In Proc. IEEE Intl. Conf. on Service Computing (SCC 2010), pp. 614-617, 2010.

[3] Fuhr, A.; Horn, T.; Riedinger, V.: Using dynamic analysis and clustering for implementing services by reusing legacy code. In Proc. 18th Working Conf. Reverse Engineering (WCRE 2011), pp. 275-279, 2011.

[4] Zhang Z.; Liu, R.; Yang, H.: Service identification and packaging in service oriented reengineering. In Proc. Intl. Conf. Software Engineering and Knowledge Engineering (SEKE 2005), pp. 620-625, 2005.

[5] Matos, C.; Heckel, R.: Migrating legacy systems to service-oriented architectures. Electronic Communications of the EASST, vol. 16, 2008.