Model-based Migration to Service-oriented Architectures

Andreas Winter
Johannes Gutenberg-Universität Mainz
Institut für Informatik,
D-55128 Mainz
http://www.gupro.de/~winter

Jörg Ziemann
Institut für Wirtschaftsinformatik im
Deutschen Forschungszentrum für künstliche Intelligenz
D-66123 Saarbrücken
http://www.iwi.uni-sb.de/

Abstract

The systematic development of service-oriented architectures in conjunction with reusing already existing software requires integration of model based software development techniques and software migration strategies. Correspondingly, we propose an approach that combines and extends methods from enterprise modelling and re-engineering for developing service-oriented architectures.

1. Motivation

Service-oriented architectures (SOA) promise flexible composition of components implementing well defined business tasks to facilitate adaptability of software systems, enabling enterprises to cope with fast changing business requirements. The development of SOA requires a profound knowledge of enterprises. *Enterprise process models* describing static as well as dynamic aspects of an organization are an established means to capture such knowledge. While static elements like business functions and organizational structures of an enterprise shape individual services, dynamic business process models drive the definition of service orchestrations. *Services* specify encapsulated functionality independent from concrete implementation issues. Services may interact to provide compound services. The functionality of services is implemented by *components* fulfilling the service specification, which can easily be composed to comprehending software systems. [Arsanjani, 2004].

On the other hand, most enterprises already run software systems that are established and time-proven. These legacy systems usually are not implemented in a service-oriented way. They provide software functionality supporting enterprises business processes by monolithic and deeply interwoven software modules. Thus, evolving legacy systems towards supporting changing business processes is limited.

Keeping legacy systems evolvable requires migration into more flexible architectures. Software migration is viewed as a transformation of software systems into a new environment, without changing its functionality [Gimnich/ Winter, 2005].

This paper discusses an approach towards migrating legacy systems to service oriented architectures by relating the intended enterprise model to the legacy software system.

2. Migration to SOA

Migrating legacy software systems to service-oriented architectures is influenced by the intended enterprise model and the legacy software system. In SOA migrations, the intended enterprise model defines the business needs of the target design (cf. [Ackermann et al., 2005]), which is determined by services and their interactions. The legacy system already contains required functionality of the service oriented target system.

In order to build flexible and reusable systems which obtain the full business potential of SOA, a sound methodology is required, taking into account existing systems and current business requirements of enterprises as well. This includes the definition of services and service interactions regarding business needs, reflected in the new enterprise model, combined with already implemented functionality based on earlier (probably no longer available) enterprise models. In a more technical vein, components implementing services have to be discovered and extracted from the legacy system and have to be transferred into the intended SOA implementation [Ziemann et al., 2006].

The presented approach on migration to service oriented architectures comprises technologies from software reengineering and business process modelling. Fig. 1 sketches a horseshoe model [Kazman et al., 1998] on SOA migration.

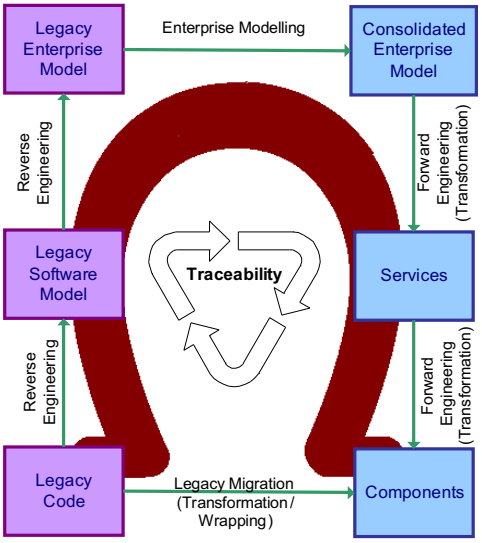

Figure 1 SOA-Migration Horseshoe

Legacy and target system are viewed on three conceptual levels showing the *code level* (Legacy Code, Components), the (platform specific) *model level* (Legacy Software Model, Services), and the *conceptual level* (Legacy Enterprise Model, Consolidated Enterprise Model). The left part presents artifacts from the legacy system. The right part presents the service oriented target system following the SOA layering [Arsanjani, 2004].

The approach follows a model-based manner. Software artefacts on all levels of abstraction are represented by appropriate models. These models provide the foundation for further *reverse engineering, enterprise modelling, forward engineering*, and *legacy migration* activities.

Reverse Engineering lifts representation of a software system on a higher level of abstraction. The main objective of the reverse engineering step in a SOA migration is to support the understanding of already existing software systems and deriving information for higher level software and enterprise models. Starting from its *source code*, re-documentation activities are used to extract a *software model*, which allows identifying services provided by the legacy system. Using additional techniques from business process modelling the software model (and other available artefacts) are used to derive an enterprise model for the legacy system. Re-documentation activities include static analysis to identify major components and dynamic analysis to determine information exchange.

Enterprise Modelling techniques are used to capture current enterprise requirements and to optimize the resulting models for a model-driven transformation to SOA. The requirements from the legacy system are summarized in the *legacy enterprise model.* Deriving the legacy enterprise models requires, next to business modelling techniques, additional requirements engineering techniques to (partially) extract those information form the *legacy software model*. Current enterprise models will be consolidated with legacy enterprise models and adapted for opportunities offered by SOA, e.g. by a stronger focus on enterprise components and new process concepts like service choreography. The outcome is a *consolidated enterprise model*. This model contains the non-functional requirements to migrate to service oriented architecture and all requirements valid for those services taken over from the legacy system.

Forward engineering comprises all activities to realize the resulting software system. In a model driven manner, the *consolidated enterprise model* represents the platform independent model, which is transformed into a platform specific model. In service oriented architectures, this platform specific model (*services*) specifies all required services and their interaction. Further transformation (and implementation) steps result in implementations by *components*. Since service implementations already exist in the legacy systems, this transformation step includes identification of those components from the legacy code.

Legacy Migration (Transformation/Wrapping) deals with transferring source code to a new environment. In SOA migration, this new environment is a set of SOA *components*. The migration activities include the discovery of service implementation in the *legacy code*, the extraction of these modules, and their adoption into the target SOA environment by wrapping or transformation.

These migration activities strongly rely on the models, serving as base for extracting abstractions or transformations. Integrating these models according to a shared metamodel provides traceability between the participating models. **Traceability** visualizes the interconnection between participating models. During migration to a service oriented architecture traceability facilitates the identification of appropriate chunks of legacy code to be extracted for certain services. Following traceability links from services in the service model, via the consolidated enterprise model, the legacy enterprise model, the legacy software model, and to the legacy code, establishes references between the implementation of services by components and the legacy implementation. From a business side, traceability is important to ensure the compliancy of IT systems with enterprise demands and for controlling purposes, e.g. for correlating the outputs of a certain component to a business function. If the resulting SOA implementation only wraps some legacy components, these traceability links also support further maintenance activities.

3. Conclusion

This proposal originates in discussions started at the Workshop "Software-Reengineering and Services", held at WI 2006 in Passau. An important result of these discussions was the adoption, that successful software migration to service oriented architectures has to bridge (business oriented) enterprise models and (technical oriented) software models. The here presented approach tries to combine enterprise modelling and software reengineering techniques to a holistic software migration method.

4. References

[Ackermann2005ERD] Ackermann E., Gimnich, R., Winter, A.: Ein Referenz-Prozess der Software-Migration (erweiterte Kurzfassung), Softwaretechnik-Trends. 25(4). 2005, 20-22.

[Arsanjani, 2004] Arsanjani, A.: Service-oriented modeling and architecture, How to identify, specify, and realize services for your SOA, IBM, 2004, http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/

[Gimnich/Winter, 2005] Gimnich, R., Winter, A.: Workflows der Software-Migration, Softwaretechnik-Trends. 25(2). 2005, 22-24.

[Kazman et al.,1998] Kazman R., Woods S., Carriere, J.: Requirements for Integrating Software Architecture and Reengineering Models: CORUM II, Working Confernce on Reverse Engineering, IEEE Computer, 1998, 154-163.

[Ziemann et al., 2006] J. Ziemann, K. Leyking, T. Kahl, D. Werth: Enterprise Model driven Migration from Legacy to SOA. In: Gimnich, R.; Winter, A.: Workshop Software-Reengineering und Services, MKWI, Passau, 2006.