Integrating a Legacy Terminal Application into an SOA

Michael Dörsam, Stefan Gründling, Thomas Langholz, Sascha Roth, Alexander Steinbrecht

Hochschule Darmstadt - University of Applied Science — Fachbereich Informatik —

Abstract: This paper is an experience report of an SOA integration project in cooperation with Software AG University Relations. An application of separating application and technical infrastructure architecture is demonstrated.

Keywords: SOA Integration, Screen Scraping, ApplinX, EntireX, IBM 3270

1 Introduction

Today’s organizations’ application landscapes still consist of terminal based legacy systems, running on reliable mainframes. Grown over the past decades, these legacy systems are involved into mission critical parts of organizations’ core businesses. Within the establishment of a service-oriented application landscape, these mission critical systems have to be integrated in a loosely coupled way.

The related project’s mission is to implement an online portal for a luxury travel agency. This portal uses different backend systems. One of these backend systems is provided by Software AG (SAG) University Relations [SAG09-4]. This backend system simulates a mainframe within a virtual machine (VM).

A legacy application written in Natural [SAG09-3] called SAG-Tours is running on this mainframe. SAG-Tours is accessible via an IBM 3270 terminal for maintaining and booking of yachts. The project’s major goal is the integration of SAG-Tours and backend systems for hotel and flight booking to provide a state-of-the-art web based travel portal for luxury travels.

Besides of these backend systems there are added-value services integrated into the portal to give customers general information about the chosen destination e.g. weather and literature recommendations. These added value services are shared from third-party providers on the Internet, e.g. Amazon or Wikipedia.
2 Project Management Issues

Since a full report of the project approach is beyond the scope of this paper, this section focuses on project meeting culture and risk management.

During weekly project meetings the participants have to answer three major questions: What have you done? What open issues do you have? What will you do next? Based on this information, we are (re-)assessing the project plan from the begin of the semester.

Furthermore, interface specifications are communicated during project meetings to move the different teams towards relatively autonomous work. Also, the architecture (see sec. 3) enhances the discussion and assures reasonable interfaces.

Plenty of new products are used to create the new portal. That requires an evaluation phase of new development tools and integration platforms which takes place on every development cycle. Each evaluation has a strategic deadline. When due, the product/technology has to be up and running without any project compromising bugs. Otherwise, the technology has to be switched to an alternative, e.g., Open Source products. This should reduce the risk of a project failure through technology risks.

3 Architecture

The architecture follows the principle of separation of concerns. Concretely, logical applications and physical implementation are separated.

The logical Application Landscape (AL) Component Architecture [EHH+08, p. 159ff] is shown in Figure 1. It hides the complexity of the actual implementation, and shows which of the different AL components are interacting.

```
Flight Service
Christopher Columbus Travel
Portal
Yacht Service
Literature Service
Weather Service
Hotel Service
Booking Process
Customer Management Service
Booking Service
```

Figure 1: Application Landscape Component Architecture

Figure 2 illustrates the Technical Infrastructure (TI) Component Architecture [EHH+08, p. 196ff] as a four-tier architecture. Client Tier components provide an interface to end users. E.g. the travel portal is accessible via a web browser. Components of the Business Process Tier implement a business processes via orchestration of services within lower
tiers. The Wrapping Tier provides Web Service interfaces. Components within this tier realize an adapter for the Application Tier. Caching algorithms are used to guarantee reliable response times and to enforce better decoupling to upper tiers. The Application Tier provides components with any kind of interfaces. These components can be used or wrapped by upper tiers. Also, foreign systems like Amazon or Wikipedia reside in this tier.

An example for the complexity reduction of the AL component architecture is given by the literature service. Literature recommendations are shown to a customer in the portal on the Client Tier. To do that, the portal calls a Web Service on the Wrapping Tier which implements an adapter and caching algorithms to provide proper response times. The data itself resides in a component on the Application Tier provided by Amazon. Another example is the service booking process in the Business Process Tier. This service orchestrates different services of the Wrapping and Application Tier.

4 Implementation

Screen scraping is a variant used to integrate applications of legacy systems by accessing its presentation. Because this variant uses the presentation, neither changes to the source code are required nor the legacy system has to go offline during deployment of the new application. In our project a proprietary product called ApplinX [SAG09-1] is used to integrate the IBM 3270 terminal of SAG-Tours.

Besides ApplinX, EntireX [SAG09-2] is used to access SAG-Tours programmed in Natural [SAG09-3] via Remote Procedure Calls (RPCs) to share the method as Web Services. The
portal itself is implemented with Java Server Faces (JSF). A Business Process Execution Language (BPEL) Engine which is part of OpenESB [Col09] is used for a travel booking process.

5 Learnings

At the end of the project we conclude the following learnings:

Discuss interface specifications as early as possible to enable teams to work autonomously. Collaborative work can be enforced due to early interface specification. Combined with loosely coupled components a high degree of autonomous development is supported.

Technical evaluation of project critical development tools and runtime environments is crucial for a successful project. Software for implementing an SOA still needs to mature. A validation of WSDLs against their XML-Schema does not guarantee any compatibility to other vendors. We exchanged the Business Process Engine of SAG because of compatibility issues while integrating Java Enterprise Edition (JEE) Web Services. Within this context, sticking to deadlines is necessary to guarantee the project’s success.

In our project, the architecture (see sec. 3) evolved over time. The AL architecture can be used to discuss new features. Combined with a mapping to the TI architecture, developers know what systems are affected by a specific new feature. The four-tier TI architecture structures the application landscape, even for future components.

References

[Col09] CollabNet: OpenESB, The Open Source ESB for SOA & Integration.  
https://open-esb.dev.java.net, last access: 10.01.2009


[SAG09-1] Software AG: ApplinX.  
http://www.softwareag.com/de/products/az/applinx.asp, last access: 10.01.2009

[SAG09-2] Software AG: EntireX.  
http://www.softwareag.com/de/products/az/entirex.asp, last access: 10.01.2009

http://www.softwareag.com/de/products/natural/default.asp, last access: 10.01.2009

[SAG09-4] Software AG: University Relations.  
http://www.softwareag.com/university, last access: 10.01.2009