Entwicklung eines Softwarewerkzeugs für die modellgetriebene Migration betrieblicher Informationssysteme

Baris Güldali, Stefan Sauer
Universität Paderborn, s-lab – Software Quality Lab Zukunftsmeile 1, 33102 Paderborn
{bguldali, sauer}@s-lab.upb.de

Perdita Löhr
TEAM GmbH, Hermann-Löns-Straße 88
33104 Paderborn
pl@team-pb.de

1 Modernisierung von Legacy-Systemen

In vielen Unternehmen und öffentlichen Verwaltungen werden betriebliche Informationssysteme eingesetzt, die auf Softwaretechnologien der 90er Jahre basieren. Diese Systeme wurden häufig auf Grundlage von 4. Generation Languages (4GL) entwickelt und folgen einer zweischichtigen Client-Server-Architektur, welche die Benutzungsoberfläche stark mit den serverseitigen Funktionalitäten und der Datenbank koppelt. Durch die enge Kopplung wird einerseits die Entwicklung von effizienten und skalierbaren Anwendungen möglich. Andererseits leidet die Wartbarkeit und Erweiterbarkeit, da die Aspekte nicht isoliert gepflegt werden können.

Moderne IT-Systeme, die auf mehrschichtigen Architekturen aufbauen, lösen die alten IT-Systeme ab. Für sie gibt es leistungsfähige Entwicklungsplattformen, die dem heutigen Stand der Technik entsprechen. Die neuen IT-Systeme sind einfacher zu warten und können leichter an die sich schnell ändernden Anforderungen angepasst werden. Das ist der Grund, warum viele Softwarefirmen zu neuen Architekturen und Softwaretechnologien wechseln. Gleichzeitig haben die Firmen ein starkes Interesse daran, ihre Investitionen zu bewahren und die mit viel Aufwand entwickelten Funktionalitäten der alten IT-Systeme wieder zu verwenden und die etablierten Entwicklungsteams zu behalten.

Die erforderliche Überführung der Legacy-Systeme in die neuen Softwaretechnologien ist aufgrund der Größe und Komplexität der Systeme mit einem hohen Aufwand verbunden und birgt erhebliche Risiken. Eine manuelle Migration erscheint in der Regel zu aufwändig und kostspielig. Durch die Entwicklung leistungsfähiger Software-Werkzeuge und Methoden, die den Migrationsprozess unterstützen und automatisieren, entstehen erhebliche Einsparpotenziale.

In diesem Beitrag beschreiben wir die Entwicklung eines Softwarewerkzeugs für die automatisierte modellgetriebene Migration von betrieblichen Informationssystemen hin zu modernen Systemen auf der Basis neuartiger Softwaretechnologien. In einem Projekt des s-lab mit einem Industriepartner soll das zu entwickelnde Produkt am Beispiel der Migration von Oracle-Technologien prototypisch realisiert werden.

2 Herausforderungen und Lösungsideen

Die Idee der automatisierten und modellgetriebenen Migration ist nicht neu und wurde bereits in früheren Arbeiten adressiert [1-4, 7]. In unserem Vorhaben legen wir den Fokus auf die nachfolgend beschriebenen Themen, die in früheren Arbeiten nicht ausreichend behandelt wurden. Dabei erläutern wir den Bezug zur modellgetriebenen Softwareentwicklung (MDSD) und beschreiben unseren Lösungsansatz.

2.1 Technologie-Unabhängigkeit

Wir haben unseren Migrationsansatz offen gegenüber Quell- und Zieltechnologien gewählt. Dies wird daran deutlich, dass wir im Migrationsprozess mehrere Modellbildungen einbringen, die sowohl von der Quell- als auch der Zieltechnologie abstrahieren. Unser Migrationsansatz kann so flexibel auf verschiedene Technologievorgaben ein- und umgestellt werden.

In ersten Phasen werden die Ergebnisse der Legacy-Analyse als fachliche, technologieunabhängige Quellmodelle aufbereitet. Auf dieser Basis werden nicht nur die Dokumentation und die Kommunikation der Ergebnisse unterstützt, sondern auch deren automatisierte Transformation in unterschiedliche Zielmodelle. Damit die Transformation in unterschiedliche Technologien erfolgen kann, müssen die Modelle folgende Voraussetzungen erfüllen: (1) Ein Quellmodell enthält neben funktionalen und strukturellen Informationen des Legacy-Systems Metainformationen über Aufbau und Kontexte des Systems. Diese werden durch die Legacy-Analyse selbst ermittelt. (2) Ein Zielmodell gibt erstens

die funktionalen und strukturellen Informationen einer Zieltechnologie vor und enthält zweitens Metainformationen, die die zu erreichenden Systemeigenschaften (z.B. Architektur, Nutzerprozesse) des zukünftigen Systems formulieren.

2.2 Nutzerzentrierte Prozessanalyse

Frühere Arbeiten sehen eine Dekomposition der technischen Anwendungskomponenten und ihre Migration in die Zieltechnologie vor [2]. Hier beschränkt sich die Neuorganisation des Legacy-Systems auf die Architektur-Ebene des Systems. In unserem Vorhaben beziehen wir Kundenexpertise mit der Legacy-Software ein und nehmen neue Anforderungen des Kunden an das Zielsystem auf. Damit nutzen wir die Chance, bei der Migration die Nutzungsprozesse zu optimieren.

Wir setzen dabei auf eine Neuorganisation der Benutzungsschnittstelle. Vor der Migration der technischen Komponenten führen wir eine nutzerbezogene Prozessanalyse (engl. User-centered Design) durch, um das Nutzungsverhalten des Anwenders in das Migrationsvorgehen mit einzubeziehen. Dafür dokumentieren wir das Endnutzerverhalten mittels Usability-Modellen [5] und ermöglichen dabei die Optimierung der Nutzer-System-Interaktion auf der Modellebene.

2.3 Legacy-Analyse nach Sichten

Wir entwickeln eine flexible Bewertungskomponente, in der rollenbasierte Auswertungen von Analysedaten erfolgen können. Unser Ansatz unterstützt die Darstellung für verschiedene Beteiligte am Migrationsprozess (z.B. Entwicklersicht, Managementsicht). Bei der Definition von Rollen und Sichten stützen wir uns auf Use-Case-Modelle, um die Nutzung der Systemfunktionen und den Informationsbedarf der Nutzerrollen zu identifizieren.

2.4 Management des Migrationswissens

Im Verlaufe jedes Migrationsschrittes werden wertvolle Erkenntnisse und Informationen über die Migration für zukünftige Migrationsprojekte oder -schritte gesammelt. Um den Migrationsprozess kontinuierlich zu verbessern, müssen diese Informationen in den Migrationsprozess wieder einbezogen werden. Unser Template-basiertes Werkzeugkonzept ermöglicht sowohl das Sammeln als auch die Wiederverwendung dieser Informationen. Unser Werkzeug erlaubt im laufenden Migrationsprozess an mehreren Stellen (z.B. GUI, Business Logik) eine gezielte Anreicherung des automatisierten Prozesses um neues Wissen. Zu jedem Zeitpunkt können Templates gezielt angepasst werden. Zudem wird der Aufbau einer themen-, technologie- und kundenbezogenen Template-Bibliothek unterstützt. Das Migrationswissen wird in Form von Modelltransformationsregeln dokumentiert.

2.5 Kundenprofile für die Code-Analyse

Eine der wichtigsten Herausforderungen der automatisierten Migration ist die Überführung von Quellcode in eine neue Zielsprache. Besonders die Entwicklungssystematik der Entwicklerteams hat entscheidenden Einfluss auf Gestalt und Struktur eines Legacy-Systems.

Wir führen deshalb eine Analyse des Legacy-Systems ein, die im Quellcode nach entwicklungstypischen Mustern sucht. Unser Ansatz basiert dabei auf dem Wissen, dass Entwicklerteams große Systeme auf der Basis gemeinsamer Programmierrichtlinien entwickeln. Die Umsetzung dieser Richtlinien im Quellcode hinterlässt Muster (ähnlich wie bei Design Patterns), die dann im Rahmen unserer Analysen gefunden und für eine gezielte Migration genutzt werden können. Auch die Definition von Antimustern [6] kann helfen, diese in der Zielsprache zu vermeiden und so die Codequalität zu verbessern.

3 Literatur

[1] H. M. Sneed, et al.: Software-Migration in der Praxis: Übertragung alter Softwaresysteme in eine moderne Umgebung, dpunkt Verlag 2010

[2] Oracle Corporation: Oracle Forms Migration Framework (OFMF), Oracle White Paper, 2010

[3] A. van Hoorn et al.: DynaMod project: Dynamic analysis for model-driven software modernization. In Proc. MDSM 2011, vol. 708 of CEUR Workshop Proc., pp. 12-13, 2011

[4] J. L. C. Izquierdo, J. Molina: Gra2MoL: A domain specific transformation language for bridging grammarware to modelware in software modernization. In Proc. MoDSE'08, 2008

[5] K. Nebe, D. Zimmermann: Suitability of Software Engineering Models for the Production of Usable Software. In Proc. EHCI/DS-VIS 2007, pp. 123-139

[6] W. J. Brown et al.: Anti-patterns. Refactoring Software, Architecture and Projects in Crisis. John Wiley & Sons, New York 1998

[7] S. Efftinge et al.: Einsatz domänenspezifischer Sprachen zur Migration von Datenbankanwendungen. In Proc. BTW 2011, LNI 180 GI 2011