Iterative Zielarchitekturdefinition in SOAMIG

Christian Zillmann, Philipp Gringel, Andreas Winter
OFFIS e.V., Escherweg 2, 26121 Oldenburg, Germany
{Christian.Zillmann,Philipp.Gringel,Andreas.Winter}@offis.de

Zusammenfassung

Software-Migrationen erfordern die Definition von Zielarchitekturen, die die Eigenschaften des neuen Architekturparadigmas weitgehend nutzen, und die die Architektureinschränkungen der zu migrierenden Systeme ausreichend berücksichtigen. Im Rahmen des SOAMIG-Projekts zur Migration in Service-orientierte Architekturen wird ein inkrementelles Vorgehensmodell zur Ermittlung einer solchen Zielarchitektur entwickelt. Der Beitrag beschreibt dieses Vorgehen und skizziert erste Anwendungserfahrungen.

1 Einführung

Informations- und Kommunikationstechniken (IuK) stelle in Unternehmen einen zu erhaltenden Unternehmenswert dar. Heute eingesetzte Softwaresysteme basieren oft auf IuK-Techniken aus den 70er und 80er Jahren. Die Pflege und Ablösung dieser Legacy-Systeme durch moderne IuK-Techniken stellt Unternehmen oft vor große Herausforderungen. Das SOAMIG-Projekt[1] zielt auf den Erhalt der Legacy-Systeme durch Transformations-basierte Migration.

2 Iteratives Vorgehensmodell

Die Architektur als Ziel der Transformation von Legacy-Systemen muss unter Erhaltung der Fachlichkeit definiert werden. Diese Zielarchitektur ist im Spannungsfeld zwischen *optimaler Zielarchitektur* und *vorhandenem Legacy-System* zu erstellen. Für das Erreichen einer möglichst optimalen Zielarchitektur, ohne dabei die zu erbringende Fachlichkeit oder das Legacy-System aus dem Fokus zu verlieren, wird ein systematisches Vorgehen zur Architekturdefinition benötigt.

Hierzu wird in SOAMIG das in Abb.1 skizzierte iterative Vorgehensmodell zur Ableitung der Architekturbeschreibung ❶ des Zielsystems entwickelt. Die Architektur wird wird hierbei schrittweise konkretisiert, bis eine detaillierte physikalisch realisierbare Architekturbeschreibung vorliegt. Regeln zur Erstellung der Architekturbeschreibung werden durch einen Architekturviewpoint ❷, d. h. ein Metamodell definiert, welches die zur Architekturbeschreibung verwendbaren Elemente und deren Beziehungen festlegt.

[1]Gefördert durch das Bundesministerium für Bildung und Forschung (01IS09017D). Vgl. auch www.soamig.de.

Um das angesprochene Spannungsfeld zwischen optimaler Zielarchitektur und vorhandenem Legacy-System zu adressieren, sieht das Vorgehensmodell ausgehend von der Fachlichkeit einen *Top-Down-Ansatz*, und ausgehend vom Legacy-System einen *Bottom-Up-Ansatz* vor.

2.1 Top-Down-Ansatz

Der Top-Down-Ansatz geht dabei von der *Fachlichkeit des Legacy-Systems* aus, die im Projekt u. a. aus der Dokumentation der Geschäftsprozesse ❸ ermittelt wurde [4]. Ebenso sind „Experten-Interviews“ mit Betreibern und Entwicklern des Legacy-Systems zur Erhebung sinnvoll. Die verwendbaren Elemente zur Beschreibung der Geschäftsprozesse werden analog zum Architekturviewpoint durch einen Geschäftsprozess-Viewpoint ❹ festgelegt. Auf Grundlage der Geschäftsprozesse ❸ wird eine erste, möglichst ideale Zielarchitektur ❺ entwickelt, die für die in SOAMIG verfolge Architektur-Migration nach SOA die Eigenschaften des Service-orientierten Paradigmas möglichst optimal ausnutzt.

Beim Entwurf der Zielarchitektur sind aber auch die Eigenschaften ❻ des Legacy-Systems zu berücksichtigen, die spätere Migrationschritte durch Transformation deutlich erschweren können. Die Überprüfung der hieraus abgeleiteten Architekturhypothesen geschieht z. B. durch „Experten-Interviews“ ❼. Sind diese Architekturhypothesen aufgrund des Zustands des Legacy-Systems nicht wirtschaftlich umsetzbar, wird durch eine Änderungsanforderung ❽ eine Überarbeitung der Architekturbeschreibung ❺ ausgelöst. Der (ideale) Architekturentwurf wird so iterativ an die Erfordernisse der Übertragung des Legacy-Systems angepasst.

In SOAMIG wurde in der ersten Version der Zielarchitektur eines Systems zum Fahrkartenverkauf eine explizite Komponente zur Preisberechnung angenommen. Diese Funktionalität wird im Legacy-System jedoch durch eine externe, im Projekt nicht zugreifbare, Komponente gemeinsam mit weiterer Funktionalität bereitgestellt. Nach der Migration des Legacy-Systems in eine SOA soll diese Kopplung weiterhin beibehalten werden. Aus den bisherigen Untersuchungen des Legacy-System ist auch zu vermuten, dass eine saubere Extraktion eines Preis-Berechnungs-Services sehr kompliziert und aufwändig wird. In der Überar-

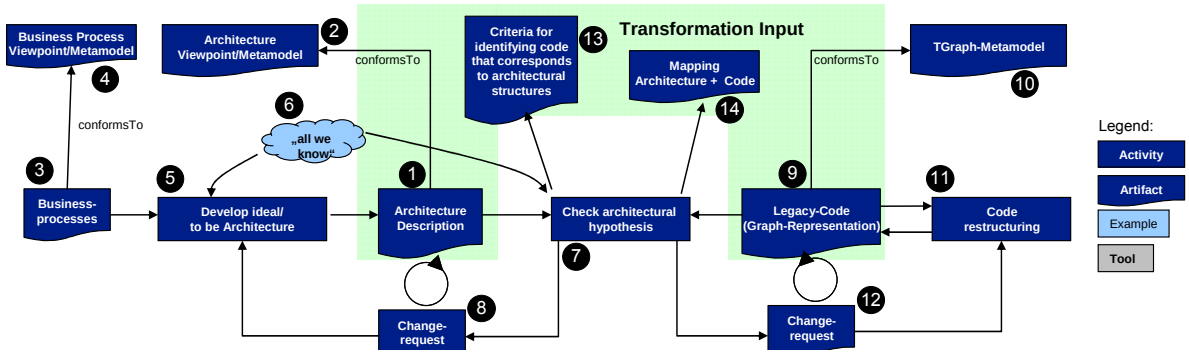

Abbildung 1: Das iterative Vorgehensmodell in der Übersicht.

beitung der Zielarchitektur wird daher die Preisberechnung in einem umfassenderen Service realisiert.

2.2 Bottom-Up-Ansatz

Aus dem *Wissen über das Legacy-System* können auch direkt Anforderungen an die Zielarchitektur abgeleitet werden. Diese werden im Bottom-Up-Ansatz bearbeitet. Das Legacy-System kann in SOAMIG mit dem FGM [1] analysiert werden und liegt als TGraph-Repräsentation ❾ [2] vor, die durch Graph-basiertes Reverse-Engineering untersucht werden kann. Hieraus abgeleitete Architekturhypothesen ❼ fließen ebenfalls über Änderungsanforderungen ❽ in die Überarbeitung der idealen Architektur ein.

In SOAMIG könnten z. B. Metriken zur Bewertung der Kopplung einzelner Methoden oder Analysen des Datenaustauschs zwischen Komponenten Indizien für die starke Verflechtung der o. g. Preisberechnung mit der Bereitstellung weiterer Reisedaten liefern, die in der Zielarchitektur eine gemeinsame Behandlung aller Reisedaten in einer Zielkomponente nahelegt.

2.3 Sanierung

Nicht erfüllte Architekturhypothesen müssen nicht notwendigerweise zur Überarbeitung der Zielarchitektur führen, sondern können auch durch die Sanierung des Legacy-Systems bearbeitet werden. In solchen Fällen wird ebenfalls eine Änderungsanforderung ⓬ ausgelöst, die jedoch eine Anpassung des Legacy-Systems bewirkt. Hierdurch kann erreicht werden, dass das Legacy-System vor der eigentlichen Migration so aufbereitet wird, dass Transformationen einfacher realisiert werden können. In diese Anpassungen können neben Code-Restrukturierungen ⓫ auch übliche Sanierungsmaßnahmen durch Refactoring einbezogen werden. Die Restrukturierung des Legacy-Systems erfordert Kriterien zur Abbildung von Legacy-Code auf Ziel-Architekturkomponenten ⓭. Durch diese Abbildungen gewinnt man auch Erkenntnisse darüber, welche Teile des Legacy-Systems in welche Komponente der Zielarchitektur zu überführen sind ⓮. Diese Informationen dienen auch als Eingaben für eine schrittweise Transformation des Legacy-Systems in eine neue Architektur und Systemumgebung [3].

Die SOAMIG-Zielarchitektur des Systems zum Verkauf von Fahrkarten wird auf klar abgegrenzten Paketen basieren. Die eigentliche Migration kann durch eine Restrukturierung des Legacy-Systems vorbereitet werden, indem vorhandene Klassen auf entsprechende Pakete aufgeteilt werden. Hierzu notwendige Abbildungen von Legacy-Code auf Paketstrukturen der Zielarchitektur werden derzeit entwickelt.

3 Ausblick

In den bisherigen Feedbackzyklen, die im Rahmen der Definition einer Zielarchitektur für das SOAMIG-Projekt durchgeführt wurden, hat sich das vorgestellte iterative Vorgehensmodell bewährt. Durch erste Iterationen wurde der initiale Architekturentwurf bereits verfeinert. Durch weitere Iterationen und die parallele Analyse des Legacy-Systems wird die aktuelle Zielarchitektur derzeit weiter konkretisiert, so dass Schritt für Schritt eine physikalisch realisierbare Architektur entsteht, in die das Legacy-System möglichst transformationsgestützt migriert werden kann.

Literatur

[1] A. Beier, D. Uhlig: Flow Graph Manipulator (FGM) - Reverse Engineering Tool für komplexe Softwaresysteme. In *Softwaretechnik-Trends*, Band 4, 2009.

[2] J. Ebert, V. Riediger, A. Winter: Graph Technology in Reverse Engineering - The TGraph Approach. In *10th Workshop Software Reengineering, Bad Honnef*, LNI P-126, 67–81, GI Bonn, 2008.

[3] A. Fuhr, T. Horn, and A. Winter. Model-Driven Software Migration. In *Software Engineering 2010, Paderborn*, LNI P-159, 69–80, GI Bonn, 2010.

[4] J. Haas, A. Fuhr, and A. Herget. Erhebung und Modellierung von Geschäftsprozessen in RAIL. Interner SOAMIG-Bericht, 2009.