Modellbasierte Migration von Unternehmensanwendungen durch Kombination eines Top-down- und Bottom-up-Ansatzes

Michael Goll
Ruhr-Universität Bochum
Universitätsstraße 151, 44801 Bochum
Email: michael.goll@rub.de

Zusammenfassung: *In diesem Artikel wird ein modellbasierter Migrationsansatz vorgestellt, bei dem Teile eines Altsystems sukzessiv migriert werden und bis zur Erstellung der letzten Komponente produktiv auf die alte Datenbank zugreifen. Das Neusystem wird für eine abstrakte Darstellung zunächst als Modell abgebildet. Zur Kompensation einer semantischen bzw. strukturellen Differenz zwischen dem Modell des Neusystems und der alten Datenbank wird eine proprietäre Komponente zur Verfügung gestellt, die beim Auslesen und Speichern von Objekten konfigurierbare Strategien ausführt. Die migrationsspezifischen Angaben werden auf Basis des Modells festgelegt. Durch diese Vorgehensweise können migrierte Komponenten frühzeitig im Produktivbetrieb eingesetzt werden und aufwendige Änderungen an den teilweise monolithischen Alt-Komponenten entfallen.*

1 Einleitung

Altsysteme sind meist geschäftskritische Systeme, die für den alltäglichen Ablauf eines Unternehmens unerlässlich sind. Teilweise sind solche Systeme mehrere Jahrzehnte im Einsatz und sie wurden in dieser Zeit kontinuierlich um zusätzliche Anforderungen (Change Requests) erweitert. Als Ergebnis können Architekturen entstanden sein, die keine klare Trennung beispielsweise zwischen der Benutzungsoberfläche und der Datenhaltung aufweisen. Diese Trennung sollte im Rahmen einer Migration wiederhergestellt werden. Weiterhin sollten zukünftige Change Requests lediglich bei den bereits migrierten Komponenten durchgeführt werden, um aufwendige Änderungen an den teilweise monolithischen Alt-Komponenten zu vermeiden. Verschärfend kommt hinzu, dass jede Änderung an dem Altsystem dazu führen kann, dass es nicht mehr lauffähig ist. Aus diesem Grund muss das Altsystem bei einer Migration unverändert in Betrieb bleiben; dies betrifft auch die alte Datenbank. Eine parallele Nutzung zweier heterogener Datenbanken ist nach [3] ein technisch komplexes Problem, daher bleibt auch nur die alte Datenbank während des laufenden Migrationsprozesses produktiv im Betrieb.

Das bisherige Altsystem kann lediglich rudimentär oder evtl. gar nicht dokumentiert sein. In diesen Fällen muss im Vorfeld eine Redokumentation durchgeführt werden. Modelle bieten hierbei den Vorteil, dass das Problem zunächst abstrakt dargestellt und diskutiert werden kann. Zudem können sie im Rahmen eines generierenden Ansatzes als Grundlage verwendet werden.

Demnach ergeben sich die folgenden Anforderungen:

1. Altsystem unverändert im Betrieb
2. Zugriff auf eine Datenbasis
3. Architekturverbesserung
4. Paralleler Betrieb
5. Modellbasiert

Die Anforderungen wurden im Kontext von Unternehmensanwendungen aufgestellt.

2 Vorhandene Ansätze

Es existieren bereits einige Ansätze in Forschung und Praxis, die die geforderten Anforderungen in Teilen erfüllen, z. B. Cold Turkey [2,3], Chicken Little [2] oder Reference Migration Process [1]. Kein betrachteter Ansatz erfüllt die Anforderungen jedoch vollständig (inhärent). Es existieren viele weitere Ansätze, die sich mit dem Themengebiet der "Migration" beschäftigen. Durch den in diesem Artikel betrachteten Kontext der Migration von Unternehmensanwendungen fallen zahlreiche Ansätze weg, die u. U. vielversprechende Aspekte, z. B. der Datenmigration, beinhalten.

3 Vorgeschlagener Ansatz

Die Grundidee des vorgeschlagenen Ansatzes wird in Abb. 1 dargestellt. Es handelt sich prinzipiell um eine Variante des Dublo-Musters [4].

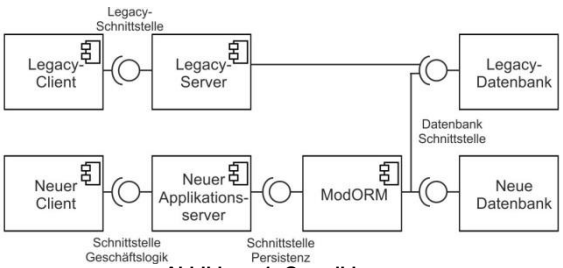

Abbildung 1: Grundidee

Das vorhandene Altsystem greift ohne Veränderung auf die alte Datenbank zu (**Anforderung 1**). Das Neusystem (migrierte Komponente des Altsystems) greift während des laufenden Migrationsprozesses über die Komponente ModORM (Modernization Object-Relational Mapping) auf die alte Datenbank zu (**Anforderung 2**). Der Zugriff über ModORM ist notwendig, da die erneuerte Komponente durch architektonische Verbesserungen semantische oder strukturelle Differenzen zur alten Datenbank aufweisen kann (**Anforderung 3**). Diese werden von ModORM unter Zuhilfenahme von *Strategien* kompensiert. Teilkomponenten des Altsystems, die bereits migriert wurden, können aus dem Produktivbetrieb genommen werden. Neue und noch nicht migrierte Teilkomponenten werden parallel eingesetzt (**Anforderung 4**). Nachträgliche Änderungen werden direkt in den neuen Komponenten vorgenommen. Im Rahmen einer Redokumentation wurde ein Modell erstellt, um eine abstrakte Darstellung zu erhalten (**Anforderung 5**). Dieses Modell kann im Rahmen eines generativen Ansatzes als Ausgangsbasis verwendet werden.
Nach Abschluss der Migration aller Komponenten werden die Daten migriert. Dieser Schritt kann ebenfalls über ModORM durchgeführt werden.
ModORM verwendet Strategien zur Überbrückung von semantischen bzw. strukturellen Differenzen. (Abb. 2).

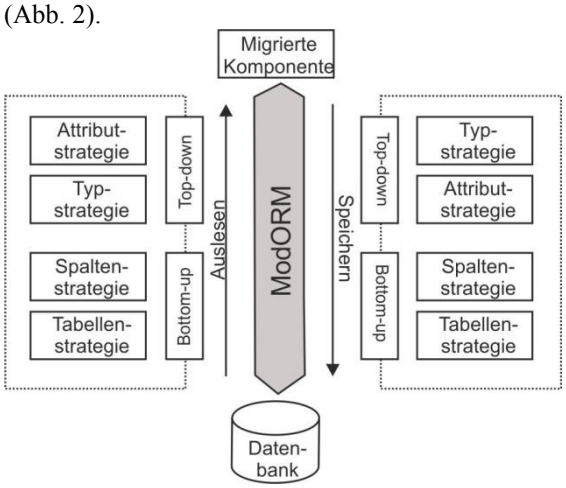

Abbildung 2: Einsatz von Strategien

Über eine Schnittstelle werden Auslese- und Speicher-Operationen zur Verfügung gestellt, die den migrierten Komponenten die Möglichkeit bieten, Objekte auszulesen und zu speichern. Zur eindeutigen Abbildung werden Strategien für jede Klasse (Modellelement) konfiguriert. Dabei werden die zwei Ebenen **Bottom-up** und **Top-down** definiert.
Bottom-up werden die Datenbankbezeichner (z. B. Spaltennamen) und Top-down die Modellbezeichner (z. B. Attributnamen) verwendet.
Die Konvertierung wird über ModORM an den Übergängen von Bottom-up zu Top-down (Auslesevorgang) bzw. von Top-down zu Bottom-up (Speichervorgang) ausgeführt. Dadurch ist eine Entkopplung der zwei Ebenen möglich.
Beim Auslesen aus der Datenbank werden zunächst alle notwendigen Informationen über die Tabellenstrategie ermittelt. Anschließend werden die Spaltenstrategien für jede Spalte durchgeführt. Nach der Konvertierung in Modellbezeichner wird zunächst die Typstrategie ausgeführt, die sich auf den gesamten Typ bezieht. Danach werden die Attributstrategien für jedes Attribut durchgeführt.
Beim Speichern ist lediglich die Top-down-Ausführung unterschiedlich: Zunächst wird die Typstrategie und anschließend werden die Attributstrategien ausgeführt. Die Tabellenstrategie sorgt abschließend dafür, dass die Daten persistiert werden.
Die Konfiguration der Modell- und Datenbankinformationen sowie der notwendigen Strategien wird in Form einer XML-Datei gespeichert. Durch den modellbasierten Ansatz können diese migrationsspezifischen Informationen über ein Plugin des verwendeten CASE-Werkzeugs aufgenommen werden.

4 Fazit

Die Komponente ModORM kann zur Entkopplung zwischen den Datenbanken und der migrierten Komponenten genutzt werden. Weiterhin sind zukünftige Erweiterungen durch Hinzufügen weiterer Strategien möglich. Anhand zweier Fallstudien konnte die Vorgehensweise praktisch erfolgreich evaluiert werden. In der ersten Fallstudie wurde ein Altsystem (Web-Architektur mit JSF) zu einer Client-Server-Anwendung migriert. Die Implementierung der Komponenten erfolgte „manuell“. In der zweiten Fallstudie wurde ein Altsystem (ebenfalls Web-Architektur mit JSF) mittels eines generativen Ansatzes in eine Web-Architektur migriert.

Quellen

[1] Ackermann, Ellen; Winter, Andreas; Gimnich, Rainer; Ein Referenz-Prozess der Software-Migration, 2005.

[2] Brodie, Michael L.; Stonebraker, Michael; DARWIN: On the Incremental Migration of Legacy Information Systems, 1993.

[3] Brodie, Michael L.; Stonebraker, Michael; Migrating Legacy Systems, Morgan Kaufmann, San Francisco, 1995.

[4] Hasselbring, Wilhelm; Büdenbender, Achim; Grasmann, Stefan; Muster zur Migration betrieblicher Informationssysteme, 2008.