Funktionsgetriebene Integration von Legacy-Systemen mit Web Services

T. Teschke[1], H. Jaekel[1],
S. Krieghoff[2], M. Langnickel[2],
W. Hasselbring[3] und R. Reussner[3]

[1] OFFIS, Bereich „Betriebliches Informations- und Wissensmanagement“
{teschke|jaekel}@offis.de
[2] Kommunale Datenverarbeitung Oldenburg (KDO)
{krieghoff|langnickel}@kdo.de
[3] Universität Oldenburg, Department für Informatik,
Abteilung Software Engineering
{hasselbring|reussner}@informatik.uni-oldenburg.de

Zusammenfassung Dieser Beitrag beschreibt die Nutzung von Web Services bei der Migration eines Legacy-Systems. Es wird eine Architektur vorgestellt, auf deren Grundlage die *Kommunale Datenverarbeitung Oldenburg (KDO)* ihre umfangreichen Legacy-Anwendungssysteme in eine moderne Softwarearchitektur integriert. Grundlage dieser Architektur ist das Dublo-Architekturmuster, das die sanfte Integration und Migration von Legacy-Anwendungssystemen unterstützt. Das betrachtete kommunale Anwendungssystem wird über Web Services in eine J2EE-Architektur integriert, wobei die zustandsabhängige Verfügbarkeit einzelner Funktionen anhand von Protokollautomaten kontrolliert wird.

1 Einleitung

Die *Kommunale Datenverarbeitung Oldenburg (KDO)* ist ein Softwarehaus und IT-Dienstleister für die kommunale Verwaltung. Angesichts des enormen Kostendrucks im kommunalen Bereich und wachsender bzw. sich änderndender Aufgaben der Verwaltungen müssen die von der KDO entwickelten kommunalen Informationssysteme flexibel und kostengünstig an neue, kundenspezifische Anforderungen anpassbar und somit langfristig nutzbar sein.

Bislang sind die Client/Server-Informationssysteme der KDO auf der Grundlage von Informix-Datenbanksystemen und der 4Js-Implementierung [Fou03] der Informix 4GL [IBM03] entwickelt worden, wobei Präsentations- und Anwendungscode in den 4GL-Programmen eng verzahnt sind. 4GL ist eine so genannte „fourth generation language“ [WW90]. Maßnahmen zur Sicherung der Konsistenz der in den Datenbankrelationen abgelegten Daten sind nicht auf Datenbankebene, sondern durch die Fachverfahren realisiert. Dies ist durch die Notwendigkeit begründet, den Betrieb auch mit alten Datenbankversionen sicherzustellen, da Kommunen aus Kostengründen nicht regelmäßig in neue Datenbanktechnologie investieren bzw. Vollwartung abschließen können.

Die KDO hat sich dazu entschlossen, diese (monolithische) 2-Schichten-Architektur mittelfristig durch eine flexiblere Mehrschichtenarchitektur zur ersetzen, die auf aktuellen Standards aufsetzt und moderne Techniken des Software Engineering berücksichtigt. Da die aktuelle Implementierung der Fachverfahren umfangreiches Domänenwissen beinhaltet und daher nicht einfach ersetzt werden kann, sollen diese Altverfahren zunächst über Web Services in die neue Architektur eingebunden und später schrittweise abgelöst werden. Das Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme (OFFIS), ein An-Institut der Carl von Ossietzky Universität Oldenburg, begleitet die KDO bei dieser Umstellung ihrer Softwarearchitektur. Die in diesem Beitrag vorgestellte Arbeit ist das Ergebnis einer Kooperation des OFFIS-Bereichs „Betriebliches Informations- und Wissensmanagement“, der Abteilung „Software Engineering“ des Departments für Informatik und der KDO.

Dieser Beitrag ist wie folgt aufgebaut: Ausgehend von einer kurzen Vorstellung des Dublo-Architekturmusters in Abschnitt 2 skizzieren wir in Abschnitt 3 drei grundsätzliche Alternativen zur Identifikation von Web Services in Legacy-Systemen. In Abschnitt 4 beschreiben wir den bei der KDO verfolgten Ansatz zur Identifikation von Web Services in den Altverfahren und deren Integration in die neue Architektur, bevor wir den Beitrag in Abschnitt 5 mit einer Zusammenfassung und einem Ausblick beschließen.

2 Sanfte Legacy-Integration und -Migration nach dem Dublo-Muster

Legacy-Anwendungssysteme unterscheiden sich von modernen Softwarearchitekturen oft darin, dass sie monolithisch aufgebaut sind und nicht zwischen unterschiedlichen Schichten für Präsentation, Anwendungslogik und Datenhaltung unterscheiden. Stand der Softwaretechnik sind heute jedoch mehrschichtige Architekturen mit klarer Trennung von Präsentation, Anwendungslogik und Datenhaltung, mit denen u. a. vereinfachte Austauschbarkeit und Anpassbarkeit von Komponenten, verbesserte Nachvollziehbarkeit fachlicher Anforderungen in der Schicht der Geschäftslogik oder erhöhte Transparenz von Datenbankaspekten angestrebt werden.

Eine sanfte Migration von monolithischen Legacy-Systemen zu modernen, mehrschichtigen Softwarearchitekturen gestattet das *Dublo-Muster* [HRJ+04]. Dublo steht für „Dual Business Logic“ und basiert im Kern auf dem Gedanken, Geschäftslogik an zwei Stellen zu halten: im Legacy-System sowie in neu eingeführten Mittelschichten. Dabei wird (neue) Geschäftslogik in einer neuen Geschäftslogikschicht implementiert und mittels eines Legacy-Adapters [GHJV96] mit der alten Geschäftslogik verbunden. Neben dem Zugriff auf alte Geschäftslogik wird dieser Adapter insbesondere auch für den Datenbankzugriff genutzt, d. h. Zugriffe der neu implementierten Geschäftslogik auf die Datenbank erfolgen ausschließlich über den existierenden Legacy-Code. Dieser Zugriff auf die Datenbank durch bestehenden Legacy-Code führt zwar zu einer Verteilung von Geschäftslogik auf zwei Schichten, hat aber den Vorteil, dass alte Geschäftslogik

zunächst weiter verwendet werden kann (um schließlich schrittweise ersetzt zu werden). Insbesondere kann so aufwändige Geschäftslogik des Legacy-Systems, die (wie bei 4GL Systemen häufig) intrinsisch mit dem Datenbenkzugriff verwoben ist, weiter genutzt werden und mehr oder weniger unverändert durch die mittleren Schichten zu einer neuen Präsentationsschicht weitergereicht werden. Im Laufe der Migration wird dann immer mehr Geschäftslogik portiert, bis der Legacy-Adapter schließlich vollständig durch einen Datenbank-Adapter ersetzt werden kann. Abbildung 1 zeigt die Struktur des Dublo-Architekturmusters.

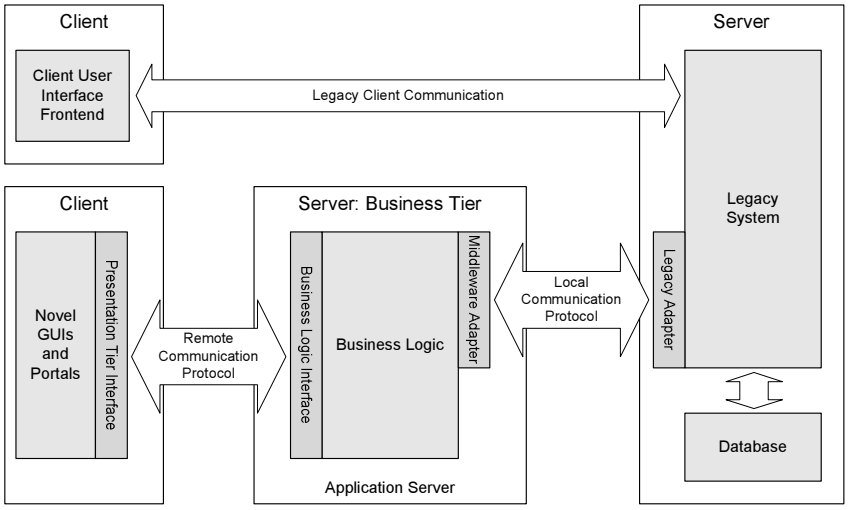

Abbildung 1. Strukturelle Sicht auf das Dublo-Architekturmuster

Im Kontext der KDO wird das Dublo-Architekturmuster eingesetzt, um einen J2EE-Server [Sha01] an das bestehende Informix-4GL-System anzubinden. Dabei werden die Legacy-Adapter über Web Services realisiert. Von der Nutzung einer standardisierten, plattformunabhängigen Technologie wie Web Services verspricht man sich insbesondere Flexibilität bei Technologiewechseln in der Datenbank- und Geschäftslogikschicht [PG03]. Alternativen zur Identifikation geeigneter Web Services werden im folgenden Abschnitt diskutiert. Die KDO wird die durch das Dublo-Muster eingeführte neue Präsentationsschicht ergänzend zum Zugriff auf neue Geschäftslogik auch für den Zugriff auf alte Geschäftslogik entwickeln, um die alte Präsentationsschicht abzulösen.

Die Verbindung zwischen Präsentationsschicht und Geschäftslogikschicht muss aus Gründen der Datensicherheit gesondert gesichert werden. Dazu kann in einer Java-Umgebung beispielsweise RMI und SSL eingesetzt werden. Im Kontext der KDO wurde diese Verbindung mittels OSCI (Online Services Computer Interface) [OSC] gemäß gesetzlichen Vorgaben gesichert. Durch Sicherung der

Verbindung zwischen Präsentations- und Geschäftslogikschicht laufen Server, die diese Schichten realisieren, in einer gesicherten Umgebung, d. h. insbesondere kann die Verbindung zwischen Geschäftslogik- und Datenbankschicht als gesichert angesehen werden.

3 Alternativen zur Identifikation von Web Services

Bei der Identifikation von Web Services (oder allgemeiner: Klassen) in Legacy-Systemen lassen sich nach [WBF97] drei alternative Ansätze unterscheiden:

1. *Datengetrieben*: Identifikation von Web Services zur Repräsentation des Objektmodells auf der Grundlage der im Legacy-System implementierten Datenstrukturen.
2. *Funktionsgetrieben*: Identifikation von Web Services zur Bündelung der für die Durchführung zentraler Anwendungsfälle relevanten Dienste auf der Grundlage der im Legacy-System realisierten Funktionalität.
3. *Objektgetrieben*: Erstellung eines Objektmodells der Anwendungsdomäne und Analyse des Legacy-Systems, um Code-Elemente zu identifizieren, die die Datenstrukturen und Funktionalität entsprechender Klassen implementieren.

Im Kontext der KDO erschien die objektgetriebene Identifikation von Web Services, die z. B. auf einem durch Entity Beans definierten Ziel-Objektmodell aufsetzen könnte, grundsätzlich nicht gangbar. Da die den Altverfahren zugrunde liegenden Relationen im Allgemeinen nicht normalisiert sind und sich daraus erhöhte Differenzen zwischen Ist- und (normalisiertem) Ziel-Objektmodell ableiten ließen, waren bei diesem Ansatz Schwierigkeiten bei der Zuordnung von Funktionen der Fachverfahren zu den Entity Beans des Ziel-Objektmodells zu erwarten.

Ähnlich wie der objektgetriebene Ansatz zielt auch die datengetriebene Identifikation von Web Services auf die Implementierung eines Objektmodells durch Legacy-Funktionen ab. Der wesentliche Unterschied zwischen beiden Ansätzen ist darin zu sehen, dass der objektgetriebene Ansatz von einem Ziel-Objektmodell ausgeht, während der datengetriebene Ansatz auf dem aktuell implementierten Objektmodell aufbaut. Beim datengetriebenen Ansatz werden geeignete Web Services im Sinne abstrakter Datentypen (ADTs) ausgehend von den im Legacy-System implementierten Datenstrukturen identifiziert. Dies können zum einen (komplexe) Datentypen, die im Quellcode der Anwendung genutzt werden, und zum anderen (relationale) Datenbankschemata sein. In der Forschung sind verschiedenartige Ansätze auf Basis von Techniken wie Clustering und Konzeptanalyse (vgl. z. B. [vDK99]) oder Datenfluss-, Kontrollfluss- und Syntaxanalyse (vgl. z. B. [KP99]) untersucht worden.

Der funktionsgetriebene Ansatz zielt im Kern auf die „Entdeckung“ von Anwendungsfällen in Legacy-Systemen ab, für die durch geeignet zusammengestellte Web Services eine Schnittstelle definiert werden soll. Dabei kann sich diese Ermittlung von Anwendungsfällen an den einzelnen Legacy-Programmen, die

die betriebliche Funktionalität des Legacy-Systems implementieren, oder an den Bildschirmmasken des Systems orientieren. Erstgenannter Ansatz wird z. B. von Erlikh [Erl02] verfolgt, der zunächst Web Services durch iterative Verfeinerung von Mengen funktional zusammengehöriger Programme identifiziert. Die APIs dieser Web Services werden anschließend durch Einsatz verschiedener Techniken des *Knowledge Mining* ermittelt. Bei dem maskenorientierten Ansatz, den Stroulia et al. in [SERS02] vorgestellen, werden durch die Analyse der Benutzerinteraktionen mit dem Legacy-System Anwendungsfälle extrahiert, für die eine neue Schnittstelle (in unserem Fall in Form von Web Services) anzubieten ist. Durch die Bestimmung der für die Durchführung dieser Anwendungsfälle genutzten Funktionen lassen sich die APIs entsprechender Web Services definieren.

4 Identifikation und Nutzung von Web Services bei der KDO

Da sich die Struktur der Implementierung der einzelnen kommunalen Fachverfahren und ihrer Funktionen primär an den unterstützten Anwendungsfällen und Geschäftsprozessen und weniger an den zugrunde liegenden Datenbankstrukturen orientiert, hat sich die KDO für einen funktionsgetriebenen Ansatz zur Identifikation und Nutzung von Web Services entschieden. Dabei erschien das von Erlikh [Erl02] vorgeschlagene Verfahren aufgrund des Einsatzes von Techniken des Knowledge Mining zu komplex. Wesentliche Gedanken des von Stroulia et al. [SERS02] untersuchten Ansatzes ließen sich dagegen auf die KDO übertragen, da in den Fachverfahren eine enge Kopplung zwischen den Bildschirmmasken und den Funktionen besteht. Sein Einsatz setzt allerdings voraus, dass die Struktur der aktuellen Masken in der J2EE-Implementierung im Wesentlichen beibehalten bleibt.

Die Identifikation der Web Services, die zur Nutzung der Altverfahren zu exportieren sind, orientiert sich an den bestehenden Fachverfahren und deren Bildschirmmasken und wird manuell durchgeführt. Für jede Maske werden dabei die Benutzerinteraktionen ermittelt, mit denen datenverarbeitende Vorgänge ausgelöst werden, die jeweils auszuführenden Funktionen analysiert und (für jede dieser Interaktionen) entsprechende Operationen eines Web Services definiert. Ein Web Service kann dabei die Operationen einer Maske oder einer funktional zusammenhängenden Sequenz von Masken umfassen. Die Parameter der Operationen eines Web Service werden aus den Feldern der jeweiligen Eingabemaske abgeleitet. Die für einen Web Service zulässigen Interaktionsfolgen (Sequenzen von Operationsaufrufen) werden durch Analyse der Abhängigkeiten zwischen Bildschirmmasken bestimmt und in Form eines endlichen (Protokoll-) Automaten spezifiziert, um die korrekte Nutzung der Dienste der Altverfahren sicherzustellen. Sind Bildschirmmasken nicht vorhanden, oder sind ihre Abhängigkeiten nicht (implizit oder explizit) spezifiziert, so können auch Verfahren zur Analyse von Geschäftsprozessen („Process Mining“) genutzt werden, um die Menge zulässiger Interaktionsfolgen zu bestimmen [Sch02]. Die explizite Angabe von zulässigen Sequenzen ermöglicht die Erkennung von unzulässigen Aufrufse-

quenzen zur Laufzeit. Der Vorteil dieser Erkennung liegt in der exakten Identifikation des ersten unzulässigen Aufrufs. Würde man diese Überprüfung unzulässiger Aufrufreihenfolgen nicht durchführen, würde das Fehlverhalten, welches durch eine unzulässige Aufrufreihenfolge verursacht wird, entweder gar nicht oder aber möglicherweise so spät entdeckt, dass eine Identifikation des unzulässigen Aufrufs nur schwer oder gar nicht möglich ist. Beschreibt man nicht nur die zulässigen Aufrufsequenzen des Adapters, sondern zusätzlich auch die von der Geschäftslogikschicht benötigten Dienste, ist eine statische Interoperabilitsprüfung der Aufrufsequenzen (Protokolle) möglich. Wird diese Prüfung schon zur Entwicklungszeit durchgeführt, können Überprüfungen der Aufrufsequenzen zur Laufzeit wegfallen, da eine statische Prüfung alle inkompatiblen Interaktionsfolgen ausschließt. Da der Aufwand einer Modellierung der von der Geschäftslogik benötigten Aufrufsequenzen als zu hoch eingeschätzt wurde, verzichtete man im beschriebenen Projekt auf statische Überprüfungen.

Ein Ansatz, der mit unserer Automatennutzung vergleichbar ist, wird bei verschiedenen Beschreibungssprachen für Web Services verfolgt. So lässt sich im Web Service Choreography Interface (WSCI) [W3C02] das beobachtbare Verhalten eines Dienstes als temporale und logische Abhängigkeiten zwischen den ausgetauschten Nachrichten beschreiben. Die Business Process Execution Language for Web Services (BPEL4WS) [BEA03] definiert eine Notation, die die temporalen und logischen Bedingungen bei der Zusammensetzung von Geschäftsprozessen aus mehreren Web Services beschreibt. Formal lassen sich Sequenzen von Web Services mit den gängigen Notationen zur Protokollspezifikation, also z.B. endlichen Automaten oder Petri-Netzen, beschreiben [BRSM03, MPP02]. Da noch nicht absehbar ist, welcher der konkurrierenden Standards WSCI oder BPEL4WS sich zukünftig durchsetzen wird, werden bei der KDO endliche Automaten zur Beschreibung der zulässigen Interaktionsfolgen eingesetzt.

Eine Voraussetzung für den Zugang zur implementierten Funktionalität über Web Services ist die Trennung der GUI-Aspekte von dem eigentlichen Anwendungscode durch Auslagerung des jeweiligen Codes in eigenständige 4GL-Funktionen. Dabei werden Interaktionen, die bislang aufgrund der Verzahnung von GUI- und Anwendungsaspekten innerhalb einer Funktion abgefragt und verarbeitet wurden, durch die Aufspaltung der 4GL-Funktionen an den Interaktionspunkten berücksichtigt. Die Web-Service-Schnittstelle kann dann auf technischer Ebene als Adapter für die resultierenden 4GL-Anwendungsfunktionen bereitgestellt werden.

Der Zugriff auf die Altverfahren der KDO über Web Services kann gemäß dem in Abbildung 2 dargestellten Sequenzdiagramm erfolgen. Dabei werden die folgenden Schritte ausgeführt:

① Aufruf eines Dienstes durch einen (Präsentations-)Client: Ein Client ruft eine Methode einer zustandsbehafteten Session Bean [DYK01] auf, die durch einen J2EE-Server bereitgestellt wird.

② Initialisierung der Session Bean: Sofern die Session Bean noch nicht initialisiert wurde, meldet diese sich zunächst bei einem „Fassaden-Server“ mit einer Kundenkennung sowie dem gewünschten Verfahren und der auszuführenden

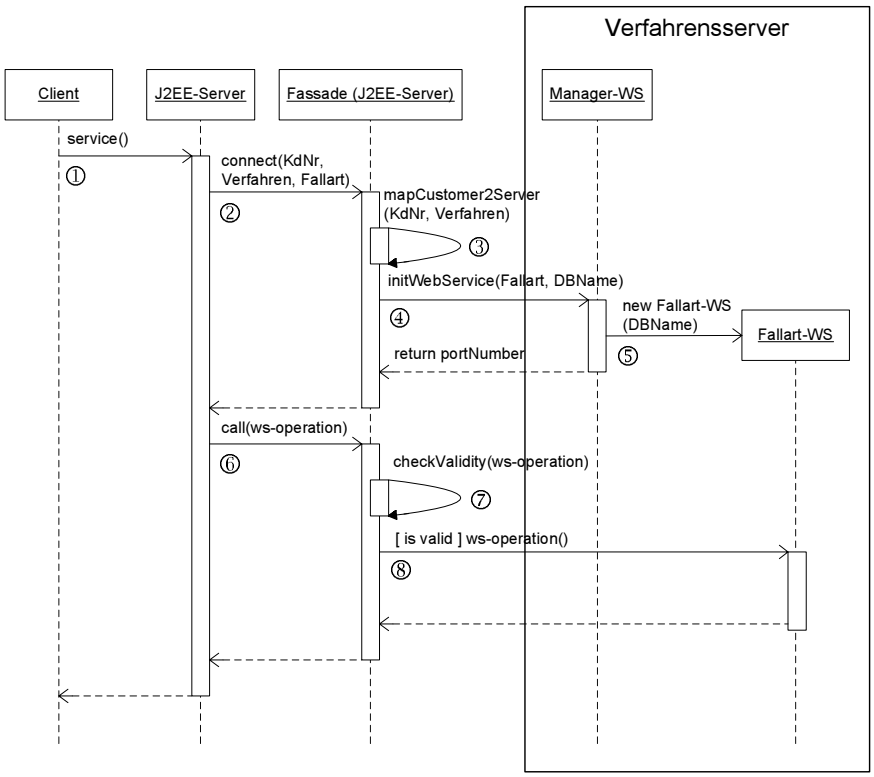

Abbildung 2. Integration von Altverfahren über Web Services

Fallart an (der Fassaden-Server bietet in Anlehnung an das gleichnamige Entwurfsmuster [GHJV96] eine abstrakte Schnittstelle zu den Altverfahren an). Ist die Session Bean bereits initialisiert, wird dieser Schritt übersprungen und direkt mit Schritt 6 fortgefahren.

③ Auswahl des Verfahrensservers: Der Fassaden-Server, dessen Dienste ebenfalls über (zustandsbehaftete) Session Beans implementiert sein können, wählt anhand der Kundenkennung und des Verfahrens einen zuständigen Verfahrensserver aus. Dabei erfolgt auch die Bestimmung der durch das Verfahren zu benutzenden Datenbank.

④ Beantragung einer Web-Service-Instanz: Der Fassaden-Server beantragt bei einem „Manager-Web-Service“, der auf dem zuvor ermittelten Verfahrensserver läuft und die Erzeugung von Web Services kontrolliert, eine Instanz des für die gewünschte Fallart zuständigen Web Services und übergibt dabei den Namen der zu benutzenden Datenbank.

⑤ Erzeugung der Web-Service-Instanz: Der Manager-Web-Service legt eine Instanz des gewünschten Web Services als neuen Prozess an (da die 4Js-Programme der Altverfahren globale Variablen nutzen, erfolgt hier eine Tren-

nung zwischen unterschiedlichen Clients durch die Erzeugung unabhängiger Prozesse). Er gibt die Nummer des Ports zurück, über den der Prozess identifiziert und angesprochen werden kann. Mit der anschließenden Speicherung dieser Port-Nummer auf dem Fassaden-Server wird die Initialisierung der Session Bean (Schritt ②) abgeschlossen.

⑥ Anforderung des Dienstes: Die zustandsbehaftete Session Bean fordert die dem clientseitig angeforderten Dienst entsprechende Operation des Web Services beim Fassaden-Server an.

⑦ Prüfung der Gültigkeit des Aufrufs: Der Fassaden-Server prüft die Zulässigkeit des angeforderten Aufrufs anhand eines Protokollautomaten, der die zulässigen Interaktionsfolgen einer Web-Service-Session beschreibt.

⑧ Ausführung der Operation: Ist der angeforderte Aufruf zulässig, wird die Operation des Web Services ausgeführt und Rückgabewerte an den Client zurückgegeben, anderenfalls wird ein Fehler signalisiert.

Aus Platzgründen verzichten wir hier auf eine detaillierte Betrachtung des Abbaus von Verbindungen zwischen J2EE-Server und Web-Service-Instanz. Inwieweit die zusätzliche Serverlast, die durch die Erzeugung einer möglicherweise hohen Anzahl von Web-Service-Prozessen entsteht, kritisch ist, muss sich im praktischem Einsatz noch zeigen.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein Ansatz zur funktionsgetriebenen Integration von Legacy-Systemen mit Web Services im Kontext kommunaler Informationssysteme vorgestellt, den wir als Weiterentwicklung bestehender Diskussionen auf dem Gebiet der Anwendungsintegration einordnen.

Im Anschluss an eine kurze Einleitung, in der die Ausgangssituation der KDO skizziert wurde, folgte eine knappe Einführung in die sanfte Integration und Migration von Legacy-Anwendungssystemen nach dem Dublo-Architekturmuster. Eine Analyse grundsätzlicher Alternativen zur Identifikation von Web Services in Legacy-Systemen bildete die Grundlage für den nachfolgend vorgestellten Ansatz zur Identifikation und Nutzung von Web Services, über den die umfangreichen kommunalen Fachverfahren in die moderne Mehrschichtenarchitektur der KDO eingebunden werden. Um die Zulässigkeit der Reihenfolge von Web-Service-Aufrufen sicherzustellen, kommen in der vorgestellten Architektur Spezifikationen endlicher Automaten zum Einsatz, auf deren Grundlage zulässige Interaktionsfolgen mit dem Legacy-System beschrieben werden.

Da die Verwendung des Dublo-Musters aufgrund der auf zwei Schichten verteilten, z. T. duplizierten Geschäftslogik einen gewissen Mehraufwand für die Pflege des Anwendungssystems impliziert, beschränkt sich die Anwendbarkeit dieses Musters auf „stabile“ Legacy-Systeme, die keine großen funktionalen oder strukturellen Änderungen mehr erfahren. Als Schwachpunkt des in diesem Beitrag vorgestellten Ansatzes muss die Beibehaltung des alten, oftmals im Laufe

der Jahre degenerierten Datenbankschemas in neuen Fachanwendungen betrachtet werden. Künftige Arbeiten betreffen daher die Konzipierung eines Migrationspfades zur vollständigen Ablösung von Legacy-Systemen und die dabei durchzuführende Evolution des Datenbankschemas. Während des parallelen Betriebs alter und neuer Geschäftlogik ist zu erwarten, dass die semantische Heterogenität zwischen alter und neuer Geschäftslogik noch weitere Herausforderungen birgt.

Literatur

[BEA03] BEA, IBM, Microsoft, SAP AG und Siebel Systems, http://www.ibm.com/developerworks/library/ws-bpel/: *Business Process Execution Language for Web Services Version 1.1*, 2003. Besucht am 18.12.2003.

[BRSM03] Berardi, D., F. De Rosa, L. De Santis und M. Mecella: *Finite State Automata as Conceptual Model for E-Services*. In: *Integrated Design & and Process Technology*. Society for Process & Design Sciences, 2003.

[DYK01] DeMichiel, L. G., L. Ü. Yalçinalp und S. Krishnan: *Enterprise JavaBeans Specification, Version 2.0*. Sun Microsystems, 2001.

[Erl02] Erlikh, L.: *Integrating Legacy Systems Using Web Services*. eAI Journal, 2002.

[Fou03] Four J's, http://www.4js.com: *Four J's Development Tools*, 2003. Besucht am 25.08.2003.

[GHJV96] Gamma, E., R. Helm, R. Johnson und J. Vlissides: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 1. Auflage, 1996.

[HRJ+04] Hasselbring, W., R. Reussner, H. Jaekel, J. Schlegelmilch, T. Teschke und S. Krieghoff: *The Dublo Architecture Pattern for Smooth Migration of Business Information Systems*. In: *26th International Conference on Software Engineering (ICSE) 2004*, Edinburgh, Schottland, Großbritannien, Mai 2004.

[IBM03] IBM, http://www-3.ibm.com/software/data/informix/tools/4gl/: *Informix 4GL product family*, 2003. Besucht am 25.08.2003.

[KP99] Kontogiannis, K. und P. Patil: *Evidence Driven Object Identification in Procedural Code*. In: *Proceedings of IEEE Conference on Software Technology and Engineering Practice (STEP '99)*, Seiten 12–21, Pittsburgh, Pennsylvania, USA, August 1999.

[MPP02] Mecella, Massimo, Francesco Parisi Presicce und Barbara Pernici: *Modeling E-service Orchestration through Petri Nets*. Lecture Notes in Computer Science, 2444:38ff., 2002.

[OSC] OSCI, http://www.osci.de: *Online Services Computer Interface*. Besucht am 18.12.2003.

[PG03] Papzoglou, M.P. und D. Georgakopoulo: *Special Section on Servuce Oriented Computing*. Communications of the ACM, 46(10):24–61, Oktober 2003.

[Sch02] Schimm, Guido: *Process Miner — A Tool for Mining Process Schemes from Event-Based Data*. Lecture Notes in Computer Science, 2424:525ff., 2002.

[SERS02] Stroulia, E., M. El-Ramly und P. Sorenson: *From Legacy to Web through Interaction Modeling.* In: *Proceedings of the International Conference on Software Maintenance (ICSM '02)*, Seiten 320–329, Montreal, Kanada, Oktober 2002. IEEE Press.

[Sha01] Shannon, B.: *Java 2 Platform Enterprise Edition Specification, v1.3.* Sun Microsystems, 2001.

[vDK99] Deursen, A. van und T. Kuipers: *Identifying Objects Using Cluster and Concept Analysis.* In: *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99)*, Seiten 246–255, Los Angeles, USA, Mai 1999. ACM.

[W3C02] W3C, http://www.w3.org/TR/wsci/: *Web Service Choreography Interface (WSCI) 1.0*, 2002. Besucht am 18.12.2003.

[WBF97] Wiggerts, T., H. Bosma und E. Fielt: *Scenarios for the Identification of Objects in Legacy Systems.* In: Baxter, I. D., A. Quilici und C. Verhoef (Herausgeber): *Proceedings of the 4th Working Conference on Reverse Engineering (WCRE '97)*, Seiten 24–32, Amsterdam, Niederlande, Oktober 1997. IEEE Computer Society Press.

[WW90] Wojtkowski, W.G. und W. Wojtkowski: *Applications Software Programming With Fourth-Generation Languages.* Wadsworth Publishing, 1990.