SOA ERFOLGREICH EINFÜHREN – VON DER IDEE BIS ZUR ERFOLGREICHEN PRODUKTIVSETZUNG

Der Wandel von einer anwendungszentrierten, heterogenen Unternehmensarchitektur mit vielen Point-to-Point-Verbindungen hin zu einer serviceorientierten Architektur (SOA) ist kein einfacher. Viel Zeit und Ausdauer müssen darauf verwendet werden. Den ersten Schritt in diese Richtung konnte die Softlab Group zusammen mit einem Kunden aus der Telekommunikationsbranche erfolgreich gehen.

Eine Vorstudie zur unternehmensweiten Systemarchitektur ergab für unseren Kunden ein großes Potenzial an Synergieeffekten durch die Einführung einer Serviceorientierten Architektur (SOA).

Als Pilotprojekt zur SOA-Einführung wurde die Ablösung einer unflexiblen und technologisch veralteten Schnittstelle zwischen Business Support Systems (BSS) und Operations Support Systems (OSS) gewählt. Die Schnittstelle überträgt Internet-Anschluss- oder Telefon-relevante Daten aus dem CRM-System zur Internet Provisioning Plattform und übermittelt den Auftragsstatus zurück an das CRM-System.

Dieses Projekt war aus mehreren Gründen ein geeigneter Kandidat für ein SOA-Pilotprojekt. Zum einen musste die Schnittstelle auf eine stabilere Technologie umgesetzt werden, da die eingesetzte Lösung oft manuelle Nacharbeiten erforderte. Zum anderen beinhaltete die Umsetzung der Schnittstelle viele Integrationsanteile, die über einen Enterprise Service Bus (ESB) umgesetzt werden konnten, sowie einen automatisierten fachlichen Prozess. Dieser fachliche Prozess wurde in der Business Process Execution Language (BPEL) modelliert und nutzte die vom ESB und weiteren Partnersystemen bereit gestellten Web Services.

Das Nachbearbeiten von Fehlern in der Schnittstelle, die abgelöst werden sollte, war aufwändig. Dies bot so viel Einsparungspotenzial, dass sich die Einführung einer neuen Softwareinfrastruktur lohnte. Die notwendigen Investitionskosten für die Hardware und Softwarelizenzen rechneten sich. Das galt auch für die Neuimplementierung der Schnittstelle basierend auf einer SOA-Infrastruktur.

In Abbildung 1 sind die beteiligten Komponenten der Schnittstelle aufgezeigt. Die Funktionalität ist kurz beschrieben.

Ein Agent gibt im Customer Relationship Management (CRM)-System Vertragsdaten ein und kann darüber auch den Status von Aufträgen abfragen.

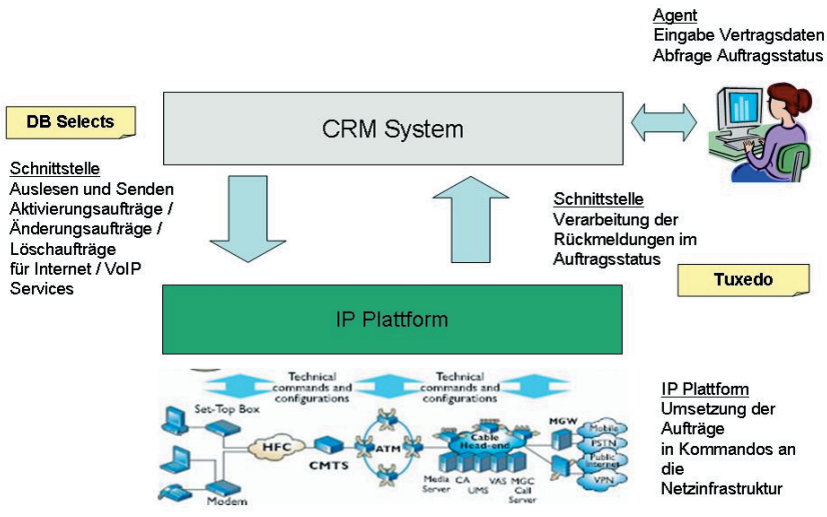

Abbildung 1: Komponenten der Schnittstelle

Alle Neuverträge, Vertragsänderungen und Kündigungen für Internet- oder VoIP-Services werden aus dem CRM-System über Datenbank-Select-Statements ausgelesen und an die IP- Plattform übertragen. Diese setzt die Aufträge in Kommandos an die Netzinfrastruktur um und meldet den Auftragsstatus zurück zum CRM-System.

Bislang wurden die fachlichen Prozesse für die Provisionierung vom CRM-System selbst unterstützt. Das CRM-System ist im vorliegenden Fall die kundenspezifische Erweiterung eines Standardprodukts, das im Telekommunikationsumfeld genutzt wird. Kundenspezifische Erweiterungen ziehen jedoch einen erhöhten Implementierungs- und Testaufwand im CRM-System nach sich. Zudem sind größere Änderungen an dem CRM-System lange im Voraus zu planen, da mit dem Produkt nur vier Releases pro Jahr möglich sind. Die Funktionalitäten, die in ein kundenspezifi-

die autoren

Harald Reinmüller
(harald.reinmueller@softlab.de)
ist SOA-Architekt und IT Consultant bei der Softlab Group in München. Sein Fokus liegt primär auf SOA-, BPM- und Java-Projekten. Er arbeitet seit mehr als zehn Jahren in den Bereichen Integration, Middleware und anderen Softwaretechnologien.

Stefan Kohlmann
(stefan.kohlmann@softlab.de)
ist SOA-Architekt und IT Consultant bei der Softlab Group in München. Seine aktuellen Tätigkeitsschwerpunkte sind SOA, EAI, BPM und Java Projekte. Er arbeitet in den Bereichen Business Prozess Modellierung, Integration, Middleware und anderen Softwaretechnologien seit mehr als 17 Jahren.

sches Release kommen, müssen langfristig zwischen dem Kunden und dem CRM-Hersteller abgestimmt werden.

Mehr Unabhängigkeit für den Kunden

Das Projekt sollte den sinnvollen Einsatz von SOA im Kundenumfeld unter Beweis stellen. Hierbei sollte eine exemplarische Umsetzung eines aktuellen Architektur-Patterns (SOA) erfolgen. Darüber hinaus sollten auch die Funktionalitäten und die Stabilität der gewählten SOA-Infrastruktur verifiziert werden. Das monolithische CRM-System sollte nur noch die entsprechenden Schnittstellen als Services zur Verfügung stellen. Fachliche Prozesse sollten aus dem CRM-System herausgelöst und unter der alleinigen Kontrolle des Kunden umgesetzt werden können – unabhängig vom CRM-System sowie den dort definierten Releasezyklen. Zudem sollte die Schnittstelle in der neuen Architektur wesentlich stabiler und flexibler sein als die bestehende Lösung. Die Nachbearbeitung von Fehlsituationen sollte wesentlich vereinfacht und somit kosteneffizienter werden.

Für den Kunden ergaben sich aus der Einführung einer SOA für diese Schnittstelle einige Vorzüge. Da die fachlichen Prozesse aus der monolithischen Applikation herausgelöst wurden, können diese unabhängig vom Lieferanten des CRM-Systems und dessen Releasezyklen erweitert und an die Gegebenheiten im eigenen Unternehmen angepasst werden. Dadurch wurde die Abhängigkeit vom CRM-System und damit auch vom Lieferanten reduziert.

Die Abbildung der fachlichen Prozesse in ausführbaren BPEL-Prozessen erhöht die Transparenz der Abläufe, da der Prozess in BPEL grafisch modelliert wird. Die grafische Modellierung erlaubt die Modifikation der Prozesse auf einer höheren Abstraktionsebene, als es mit den üblichen Implementierungssprachen möglich wäre. So ist die Modifikation eines Prozesses leichter durchführbar. Zudem sind die Abläufe, die von einer Änderung betroffen sind, leicht zu erkennen und entsprechend isoliert testbar.

Die Kommunikation zum CRM erfolgt über Tuxedo-Schnittstellen. Der direkte Aufruf dieser Schnittstellen ist aufgrund der Technologie mit einem hohen Einarbeitungsaufwand verbunden. Durch die Kapselung der Schnittstellen in ESB-Services wird die Wiederverwendbarkeit der Schnittstellen erhöht, die vom CRM-System bereitgestellt werden. Denn nun erfolgt der Aufruf der Schnittstellen über den ESB. Der ESB stellt alle Services in Form von Web Services zur Verfügung, die sich sehr leicht in moderne Entwicklungsumgebungen integrieren lassen.

Projektvorgehen im Überblick

PoC: In einem umfangreichen Proof-of-Concept (PoC) wurde ein geeigneter SOA-Infrastruktur-Lieferant ausgewählt. Die SOA-Infrastruktur sollte dabei aus den Komponenten ESB, BPEL-Engine, Rules-Engine und einer Service-Registry bestehen. Der PoC umfasste Testfälle sowie Anbieterstatements. Diese waren gebündelt in ein „Evaluation Framework", das die kurz- und mittelfristigen Anforderungen unseres Kunden enthielt. Alle drei getesteten Hersteller durchliefen den PoC erfolgreich. Mit dem Punktsieger Oracle und dessen SOA-Suite konnte dann das eigentliche Pilotprojekt gestartet werden.

Architektur: In der Konzeptionsphase wurde die Architektur verfeinert, die in groben Zügen schon im PoC erstellt wurde. Die Beschreibung der Architektur unterteilt sich in verschiedene Sichten. Die konzeptionelle Sicht bietet einen Überblick über die betroffenen Systeme und deren Schnittstellen. In der funktionalen Sicht werden die fachlichen und technischen Komponenten auf die Tiers (Client, Process, Service, Resource) verteilt. Die Ausführungs- und Verteilungssicht beschreibt eine Sicht auf das System zur Laufzeit. Die Sicht zeigt zudem, wie die Komponenten der funktionalen Sicht diesen Elementen zugeordnet sind und wie diese Elemente untereinander kommunizieren.

Prozess- und Servicemodellierung: Ein weiteres sehr wichtiges Thema in der Konzeptionsphase war die Modellierung der Prozessabläufe. Die fachlichen und technischen Prozesse, die zu implementieren waren, wurden grafisch mit der Business Process Modeling Notation (BPMN) beschrieben. BPMN erwies sich als eine sehr mächtige und intuitive Prozessbeschreibungssprache, die allen unserer Anforderungen genügte. Aus den Aktivitätselementen, die in den Prozessen verwendet wurden, konnten Servicekandidaten extrahiert werden. Diese wurden im Servicedesign um Schnittstellen- und Verhaltensbeschreibungen erweitert.

Abbildung 2 zeigt den vereinfachten Geschäftsprozess in BPMN.

Modell: Als kanonisches Klassen-Modell wurde das Shared Information and Data (SID)Modell des TeleManagement Forums verwendet. SID ist eine standardisierte Datenrepräsentation. Diese definiert die Entitäten und deren Kommunikation untereinander. Diese Zusammenstellung von Informationsmodellen wird genutzt, um alle beteiligten Komponenten (z. B. „Kunde", „Bestellung", „Netzwerkdienst", „Konfiguration", etc.) in einem OSS-System zu beschreiben. Das SID-Modell liegt als eine Sammlung von XML Schema Definitionen (XSDs) vor, von denen konkrete Nachrichtenelemente abgeleitet werden müssen. Diese Nachrichtenelemente werden in den WSDLs der Web Services referenziert.

Das Klassenmodell in Abbildung 3 zeigt die SID-Basis-Entitäten Service und Resource mit deren Beziehungen untereinander.

Service Implementierung: In der Implementierungsphase galt es in einem ersten Schritt, die im Design beschriebenen Services zu erstellen. Dabei musste insbesondere das CRM-System integriert werden. Lesende Zugriffe sollten als direkte DB-

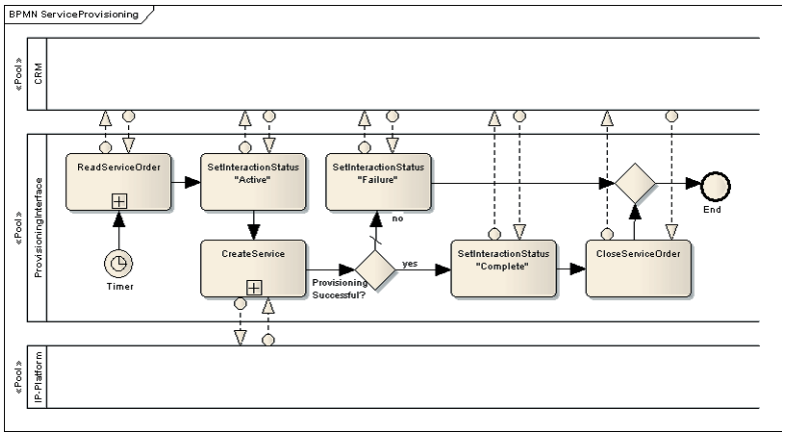

Abbildung 2: vereinfachter Geschäftsprozess in BPMN

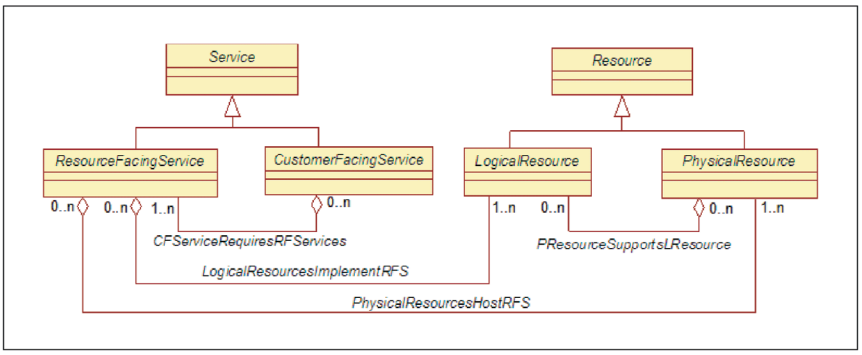

Abbildung 3: SID-Basis-Entitäten Service und Resource

Zugriffe erfolgen, für schreibende Zugriffe sollten bereits existierende Tuxedo-Services wieder verwendet werden. Hierbei leistete der ESB gute Integrationsdienste. Es gelang ohne Probleme, die DB-Zugriffe (Select-Statements) sowie die bestehenden Tuxedo-Services – mittels der mitgelieferten Adaptoren – als Web-Service zu kapseln. Die CRM-Seite war somit integriert. Die Netzverwaltungsseite stellte zu dieser frühen Implementierungsphase bereits Web-Services zur Verfügung, und zwar mit einer Dummyimplementierung in einer Testumgebung. Somit war der Weg frei, um all diese Services mittels BPEL zu ablauffähigen Prozessen zu orchestrieren.

BPEL Implementierung: Als Vorlage für die Prozessimplementierung dienten die BPMN-Diagramme, die es im BPEL-Designer umzusetzen galt. Eine Generierung von BPEL-Code aus dem BPMN-Modell wäre hier wünschenswert gewesen. Dieses Feature unterstützte das von uns eingesetzte Produkt zur BPMN-Modellierung jedoch nicht. Mancher kleine BPMN-Prozess wurde dann in BPEL umfangreicher als ursprünglich gedacht. Der Grund: In BPEL findet viel Transformations- und Fehlerbehandlungslogik statt, die wir nicht im den BPMN-Diagrammen modelliert haben.

Abbildung 4 zeigt einen Ausschnitt aus einem BPEL-Prozess im BPEL-Designer.

Deployment: Durch ein ausgefeiltes Ant-basiertes Deployment konnten alle BPEL-Prozesse sowie deren abhängige Sub-BPEL-Prozesse und ESB-Services direkt auf der Zielumgebung (Entwicklungs-, Integrations- oder Produktivumgebung) installiert werden.

Um den Überblick über die Prozess- und Service-Abhängigkeiten mit Blick auf deren Versionen zu behalten wurde ein Ant-Task programmiert. Dieser generierte automatisch bei jedem Deployment Abhängigkeitsgraphen. Abbildung 5 zeigt einen solchen Graphen. Dabei stellt jede Box einen Prozess bzw. Service dar (grün: public, rot: private, weiß: ESB).

Abbildung 4: BPEL-Prozess im BPEL-Designer

Highlights und Herausforderungen im Projekt

Die konzeptionellen Phasen gestalteten sich problemlos, was nicht anders zu erwarten war. Herausforderungen und Highlights gab es dann jedoch reichlich bei der Implementierung der Services und Prozesse. Einige Highlights der SOA-Implementierung mit der Oracle SOA Suite: die einfache Web Service-Generierung, die regressionsfähigen Unit-Tests sowie die Transparenz der Abläufe der einzelnen Prozessinstanzen.

Keine Codierung durch Web Service Creation Wizards: Um bestehende Legacy-Anwendungen zu integrieren, gibt es viele mitgelieferte Adaptoren. Diese erzeugen mittels Wizards fertige Web Services, die auf dem ESB deployed werden können. Dadurch war es uns möglich, die Oracle-Datenbank und die bestehenden Tuxedo-Services schnell und einfach zu generieren.

Unit Tests für BPEL-Prozesse: Die Oracle SOA Suite bietet die Möglichkeit, jeden BPEL-Prozess mit Testfällen zu versehen. Das erspart die manuelle Arbeit des Testens nach jeder Änderung an einem BPEL-Prozess. Dabei können Nachrichten emuliert und mit Value-Asserts Variableninhalte validiert werden – an beliebigen Stellen im Prozess. Zudem ist die Kontrolle des Programmflusses mit „Activity executed"-Asserts möglich.

Transparenz durch Instance-Flow: Ein sehr schönes Feature bringt die BPEL-Console mit der Instance-Flow-Darstellung mit. In dieser Ansicht wird eine Prozess-Instanz zur Laufzeit visualisiert. Dadurch ist völlig transparent, welchen Weg diese Instanz durch den Prozess genommen hat. An jeder Aktivität der bereits durchlaufenen Aktivitäten lassen sich die Variableninhalte anzeigen.

Abbildung 6 zeigt einen Ausschnitt aus einem Instance-Flow mit einem Auszug der Variableninhalte der empfangenen Input-Nachricht.

An dieser Stelle soll auch auf zwei Herausforderungen aus dieser Implementierung eingegangen werden, die von der gewählten SOA-Suite nicht im erforderlichen Maße unterstützt werden bzw. um die in der BPEL Engine vorhandenen Tuningmöglichkeiten voll auszunutzen.

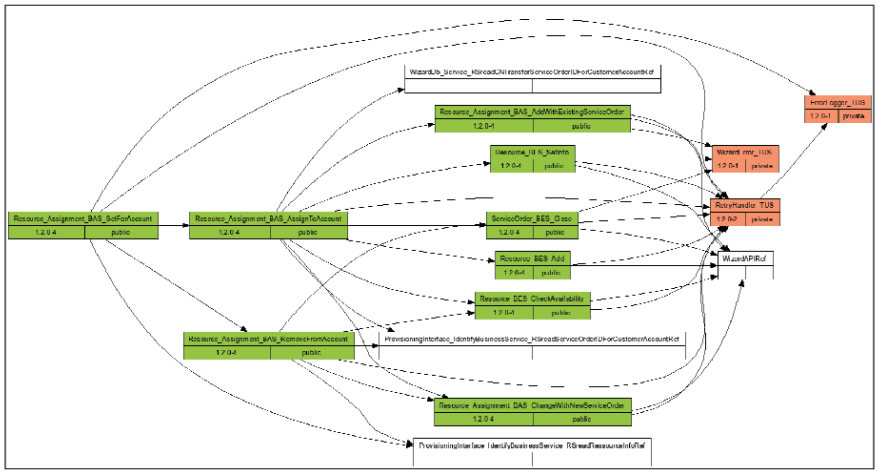

Abbildung 5:Abhängigkeitsgraph eines Deployments

Retry-Handling: Wir konnten nicht davon ausgehen, dass unsere Schnittstellenpartner immer verfügbar sind. Deshalb waren wir darauf angewiesen, einen intelligenten, automatisierten Retry-Mechanismus zu entwickeln. Die BPEL-Engine bringt zwar einen einfachen Retry-Mechanismus mit. Dieser erfüllte unsere betrieblichen Anforderungen jedoch nur unvollständig. Aus diesem Grund wurde bei jedem BPEL-Invoke eines Partnerlinks der Fehler „RemoteFault" abgefangen, um auf Kommunikationsfehler mit einem definierten Ablauf reagieren zu können. Dieser Ablauf sah eine unendliche Wiederholung vor, die ersten drei Male im Ein-Minuten-Intervall. Nach drei Fehlversuchen gehen wir von einem tatsächlichen technischen Fehler beim Aufruf des Partnerlinks aus und erzeugen eine entsprechende Eskalationsnachricht auf einem definierten Fehlerkanal an den Applikationsbetrieb. Zudem wird unser Hauptprozess deaktiviert. Weitere Aufrufe des Partnerlinks erfolgen ab jetzt im Zehn-Minuten-Intervall. Beim ersten erfolgreichen Aufruf des Partnerlinks wird der Hauptprozess wieder aktiviert. Mit dem Deaktivieren und Aktivieren unseres Hauptprozesses konnten wir im Fehlerfall Last von der BPEL-Engine nehmen, da während der Deaktivierung des Prozesses keine neuen Instanzen erzeugt werden.

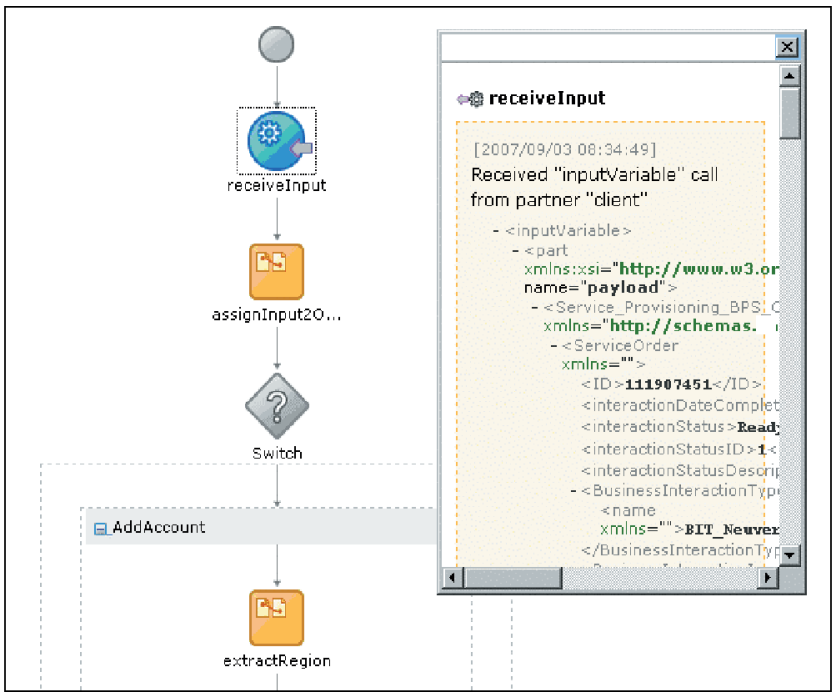

Abbildung 6: Ausschnitt aus einem Instance-Flow

Dieser intelligente Retry-Handler erlaubt beispielsweise eine Downtime von externen Systemen, ohne dass die SOA-Suite mit herunter gefahren werden muss. Das führt zu einer effektiven betrieblichen Entkopplung der SOA-Infrastruktur von den externen Systemen.

Hohe Last und Thread-Tuning: Eine weitere große Herausforderung bestand darin, die komplette SOA-Umgebung zu tunen, um den hohen Last-Anforderungen von mehreren hunderttausend langlaufenden Prozessinstanzen gerecht zu werden. Hierzu wurde an der einen oder anderen Thread-Schraube gedreht. Auch um kleine Design-Änderungen bei den BPEL-Prozessen kamen wir im Zuge der Performance Optimierung nicht herum.

SOA in den Köpfen

Die Umsetzung der Schnittstelle in eine SOA-Architektur war sehr erfolgreich. Sie erfüllte die fachlichen und technischen Anforderungen, die im Vorfeld an das Pilotprojekt gestellt wurden. Die Produktivsetzung der Schnittstelle verlief für den Anwender vollkommen transparent und ohne große Anlaufschwierigkeiten – obwohl die Softwareinfrastruktur ausgetauscht und die fachlichen Prozesse vollkommen neu implementiert wurden.

Durch den Einsatz von ESB-Services wurde eine erhöhte Wiederverwendung bestehender Services erreicht. Durch die Orchestrierung der Web Services mit Hilfe von BPEL-Prozessen ließ sich die kundenspezifische Logik aus dem CRM-System herauslösen.

Fachliche und technische Probleme können gelöst werden. Auch die Produktreife der SOA-Umgebung nimmt immer mehr zu. Ohne den SOA-Gedanken in den Köpfen der Mitarbeiter geht es jedoch nicht. Hier konnte unser Projekt zwar den ersten wichtigen Schritt in die richtige Richtung gehen, damit ist es jedoch noch lange nicht getan. Erfahrene SOA-Architekten, SOA Governance und Rückhalt vom Management sind weitere wichtige Erfolgsfaktoren für eine erfolgreiche SOA-Einführung in einem Unternehmen. ■