mehr zum thema:
www.sdm.de/de/angebot/loesungen/soa/index.html

SOA-TRANSFORMATION VON LEGACY-ANWENDUNGEN

Der in der Vergangenheit investierte Entwicklungsaufwand und das umgesetzte Prozesswissen machen die bestehende Anwendungslandschaft für ein Unternehmen zu einem wertvollen Investitionsgut. Die Einführung einer Service-orientierten Architektur (SOA) kann deshalb nicht die Ablösung und Neuentwicklung aller bisherigen Anwendungen zum Ziel haben, sondern nur ihre Erhaltung auch unter den veränderten Anforderungen und der veränderten Nutzung. Der Artikel stellt mögliche Lösungsmuster und ihre Anwendung auf Grundlage einer Typisierung der Altsysteme vor.

der autor

Oliver F. Nandico
(E-Mail: oliver.f.nandico@sdm.de) ist Principal Consultant in der IT-Beratung der sd&m AG und arbeitet als „IT Enterprise Architect". Intern wirkt er bei sd&m in der Expertengruppen „Enterprise Integration/ Service Oriented Architecture" mit.

Wenn Gartner den Begriff „Service-orientierte Architektur" 2006 nicht mehr im Bericht zu „Emerging Technologies" (vgl. [Pet06]) aufführt, kann das zweierlei bedeuten: Entweder hat sich dieser Ansatz als Sammlung leerer Versprechungen erledigt oder aber er ist inzwischen so etabliert, dass er nicht mehr eine aufkommende, sondern vielmehr eine bewährte Technik ist. Eine ganze Reihe von Seminaren, umfangreiche Literatur und – am wichtigsten – die Erfahrungsberichte von IT-Entscheidungsträgern beweisen: *Service-orientierte Architektur (SOA)* ist ein allgemein anerkanntes Prinzip für die Gestaltung von Anwendungslandschaften in Unternehmen.

Jeder Befürworter des Einsatzes von SOA, jeder IT-Verantwortliche, der von seinen guten Erfahrungen mit der Umsetzung dieses Gestaltungsprinzips berichtet, betont dabei zu Recht, dass die veränderte Ausrichtung der Anwendungslandschaft einen nachvollziehbaren Nutzen zeigen muss. Dieser besteht unter anderem in einer deutlich gestiegenen Flexibilität in der Anwendungslandschaft, der schnelleren Umsetzung neuer fachlicher Funktionen und dem geringeren Aufwand bei ihrem Test und ihrer Einführung.

Die besondere Herausforderung für den IT-Unternehmensarchitekten besteht darin, den Umfang und die Tiefe der Umgestaltung der Anwendungslandschaft in Richtung der SOA festzulegen. Er beachtet dabei Wirtschaftlichkeit, fachliche Anforderungen und technische Möglichkeiten und realisiert so auch den versprochenen Nutzen. Der dadurch festgelegte Zeithorizont und der maximale Gesamtaufwand für ein Programm zur Umsetzung einer SOA grenzen die Möglichkeiten für die Entwicklung wirklich neuer Services ein. Weiter stellen die bestehenden Anwendungen für ein Unternehmen ein Investitionsgut von hohem Wert dar – sowohl als fachliche Lösung als auch als materialisierter Realisierungsaufwand (vgl. [Hes04]). Eine SOA unter wirtschaftlichen Randbedingungen einzuführen, bedeutet also vor allem, die fachlichen Funktionen der bestehenden Anwendungen als Services verfügbar zu machen.

Für die SOA ist auch die Technik der Servicekommunikation festgelegt, z. B. über Web-Services. Anwendungen, die nicht schon ihre fachlichen Leistungen als Services auf dieser technischen Basis anbieten, sind Alt- oder Legacy-Anwendungen im Sinne der Einführung einer SOA. Bei den meisten Organisationen gilt dies für den überwiegenden Anteil der Anwendungen.

Die Einführung einer SOA ist damit vor allem ein Programm zur Renovierung der bestehenden Anwendungslandschaft und zur Transformation der bestehenden Legacy-Anwendungen. Regeln für eine SOA von hoher Qualität, wie sie in [Hes06] formuliert sind, helfen natürlich, Services zu definieren. Um sie in einer Umgebung umzusetzen, die überwiegend von Altanwendungen geprägt ist, benötigt der IT-Architekt Muster für die Transformation von Legacy-Anwendungen.

Klassifizierung von Altanwendungen

Die Muster für diese Transformation stützen sich auf eine Klassifizierung der Altanwendung. Diese richtet sich in einer SOA nach der Art der Abweichung einer Anwendung vom Idealzustand. Auf diese Klassifikation stützen sich die in diesem Artikel dargestellten Lösungsmuster.

Die Anwendung bietet nur speziell definierte Schnittstellen, keine Services

Eine Anwendung, die keine Services anbietet und auch keine nutzt, kann nicht in eine SOA integriert werden. Speziell definierte Schnittstellen genügen typischerweise weder dem grundsätzlichen Anspruch an Services als abgeschlossene fachliche Leistung, noch folgen sie der Technik, die die jeweils definierte SOA für die Servicekommunikation festlegt.

Ein Beispiel hierfür ist die Ausführung von Schnittstellen als Zugriff auf eine gemeinsam genutzte Datenbank oder die Datenübergabe per erzeugter und genutzter Datei. Auch wenn die Anwendung über eine Programmierschnittstelle verfügt, diese aber nur sehr elementare, also fachlich nicht vollständige Funktionen anbietet oder die Nutzung verschiedener Funktionen in einem Sitzungskontext erwartet, ist das kein Angebot von Services.

Diese Klasse von Anwendungen – ohne Angebot definierter Services – ist der Standardfall für eine notwendige Transformation.

Die Anwendung bietet nur fachlich eingeschränkte Funktionen

Die Informationen, die Art der Verarbeitung oder das fachliche Modell einer Anwendung können so eingeschränkt sein, dass sich für eine Anwendung keine sinnvollen Services definieren lassen. Anwendungen, die – häufig provisorisch – für bestimmte spezielle Anwendungsfälle entstanden sind, können von ihrer fachlichen Struktur her keine allgemeingültigen Services anbieten. Häufig entsteht dieses Problem auch, weil sich nach der Entwicklung der Anwendung die fachliche Sicht erweitert hat. Nicht notwendiger-

weise bedeutet die fachliche Einschränkung einer Anwendung, dass ihr funktionaler Umfang gering ist. Auch und gerade in funktional umfangreichen Anwendungen sind bestimmte Anforderungen häufig nur auf den Zweck der Gesamtanwendung hin zugeschnitten und so eben nur eingeschränkt umgesetzt.

Als Beispiel kann etwa eine funktional umfangreiche Vertriebsanwendung als „Termin" ein Treffen von einem zuständigen Vertriebsmitarbeiter mit einem möglichen Kunden sehen. Erweitert sich jetzt die fachliche Sicht und versteht unter einem Termin ein Treffen einer beliebigen Anzahl von Personen, die auch alle Mitarbeiter des Unternehmens sein können, dann kann dafür die fachlich eingeschränkte Anwendung ohne Änderung keine Services anbieten.

Technische Lücke zwischen der Anwendung und der technischen Plattform für die SOA

Eine technische Lücke zwischen einer Anwendung und der technischen Plattform für die SOA besteht dann, wenn die Technik der Servicekommunikation von der in der Anwendung verwendeten Technik so abweicht, dass eine Integration nicht möglich ist. Die isolierte Betrachtung dieses Aspekts geht davon aus, dass die Modellierung von Services in der Anwendung möglich und umgesetzt ist, also nur eine technische Lücke besteht.

Ein Beispiel mag diesen Punkt illustrieren: Ist etwa für die Servicekommunikation die ausschließliche Nutzung von Web-Services vorgesehen, dann ist für eine Anwendung, die auf dem Einsatz und der Nutzung von COBOL und CICS basiert, eine technische Lücke zur Integrationsplattform zu konstatieren. In einem Ansatz „Serviceorientierung am Host" mit Einsatz entsprechender Mittel existiert diese Lücke nicht.

Anwendung mit zu vielen fachlichen Funktionen

Anwendungen mit einer Vielzahl an fachlichen Funktionen führen zu sehr komplexen und nur schwer beherrschbaren Softwaregebilden. Eine Anwendung, die viele fachliche Funktionen implementiert, schafft Abhängigkeiten zwischen diesen, auch wenn sie fachlich nicht begründet sind. Die mit der Anzahl der fachlichen Funktionen exponentiell steigende Anzahl der Abhängigkeitsbeziehungen führt zwangsläufig zu einer überbordenden Komplexität. Daneben sorgt die gemeinsame Umsetzung einzelner Funktionen auch für Seiteneffekte: Die gemeinsame Nutzung einzelner Codeteile, die Nutzung von Vererbung und Spezialisierung in der Umsetzung und die umfassende Verwendung technischer Basiskomponenten schafft auch in gut strukturierten und modularisierten Anwendungen Abhängigkeiten in der Form, dass Änderungen für eine fachliche Funktion ein verändertes Verhalten in einer anderen fachlichen Funktion zur Folge haben können.

Anwendungen, die auch nach enger fachlicher Sicht viele fachliche Funktionen implementieren und sehr unterschiedliche Informationen verarbeiten, tendieren dazu, die Funktionsvielfalt weiter auszubauen. Der Grund dafür ist, dass für neue fachliche Anforderungen wesentliche Ansätze schon in der Anwendung gegeben sind. Die neuen Anforderungen sind am schnellsten und – unter kurzfristigen wirtschaftlichen Überlegungen – mit dem geringsten Aufwand in der bereits funktional umfangreichen Anwendung zu erfüllen. Das einschlägige „Anti-Pattern" ist hier *Feature Creep*, also die Forderung nach immer neuen Funktionen ohne geordneten Prozess zur Steuerung der Änderungen.

Beispiele von Anwendungen, deren Größe zu nur schwer beherrschbarer fachlicher Komplexität führen, finden sich fast immer in der realen Anwendungslandschaft. Ein häufiger Fall ist die Umsetzung eines ERP-Ansatzes in einem Unternehmen. Das führt typischerweise zur Planung und zumindest teilweisen Realisierung von Systemen, die die gesamte Prozesslandschaft des Unternehmens unterstützen sollten.

Vermischung technischer Integration mit fachlichen Funktionen

Komponenten sind so zu definieren, dass technische und fachliche Aspekte getrennt sind (vgl. [Sie03]). Dies schafft die Unabhängigkeit zwischen beiden und sichert die Flexibilität und Zukunftssicherheit der Anwendung. Insbesondere fordert der SOA-Ansatz, fachlich definierte Services unabhängig von Komponenten der technischen Infrastruktur zu Realisierung der Servicekommunikation zu halten. Das heißt dann, dass eine anwendungsübergreifende Geschäftsprozesssteuerung, technische Rahmen zur Integration an der Benutzungsoberfläche, Anwendungsportale und auch die Technik zur Verteilung von Nachrichten als Basis der Servicekommunikation nicht Teil einer fachlichen Anwendung sind (vgl. [Hes06]).

Ein Beispiel für einen der Trennung von Anwendungen von der technischen Infrastruktur entgegenstehenden Ansatz ist eine „Daten-Drehscheibe". Das ist eine Anwendung, die Daten sammelt, prüft, aufbereitet und anreichert und an andere Anwendungen verteilt. Die technisch-infrastrukturelle Funktion des Sammelns und Verteilens von Daten ist mit der fachlich bestimmten Funktion der Prüfung, Aufbereitung und Anreicherung der Daten vermischt. Ursache hierfür ist häufig, dass die Gestalter der Anwendungslandschaft die fachliche Anwendung typischerweise als Bestandssystem für die Daten geplant hatten, dieses System aber auch Mängel in der Infrastruktur kompensieren musste, z. B. das Fehlen einer allgemeinen Technik für den Transport von Nachrichten.

Auch Anwendungen mit zu vielen fachlichen Funktionen tendieren dazu, Aspekte der technischen Infrastruktur umsetzen zu wollen. Auf diese Weise wird die Integration in eine Anwendungslandschaft zur Integration in eine Anwendung, was einem der Grundsätze jedes ordentlichen Softwareentwurfs entgegensteht: der Trennung der Verantwortlichkeiten (*Separation of Concerns*).

Batch-Ketten

Batch-Ketten, die als gestaffelt arbeitende Filter auf einen großen Datenbestand wirken, verschließen sich der Einbindung in eine SOA nahezu vollständig. Typischerweise bieten sie auch keine aufrufbaren Schnittstellen nach außen an. Da sie auch nur während ihres Laufes verfügbar, aber auch dann nicht wirklich über Services nutzbar sind, sind solche Batch-Ketten ein Stolperstein für jede Form der Servicekommunikation. Dadurch, dass diese Batch-Läufe auch nur in periodischen Abständen zur Ausführung kommen, beeinträchtigen sie in jedem Fall die Aktualität der in der Umgebung gehaltenen Gesamtinformation. Batch-Ketten verhindern Absichten in Richtung „Realtime Enterprise", die auch eine Motivation für die SOA sind.

Schon aus historischen Gründen – ursprünglich war ja die betriebliche Datenverarbeitung vor allem ein Netz von Batch-Programmen – sind sie in realen

Anwendungslandschaften weit verbreitet. Ihr Anwendungsfeld ist die periodische Massenverarbeitung, zum Beispiel die Erzeugung großer Mengen von Monatsrechnungen und anderen Informationen für Kunden. Die Entwickler solcher Systeme führen als Begründung häufig an, dass sie bestimmte Ressourcen, etwa volumenstarke Drucksysteme, gleichmäßig und optimal auslasten wollen. Wo und inwieweit dieser Punkt die Anforderungen nach Aktualität aufwiegt, wird der Gestalter einer serviceorientierten Umgebung untersuchen.

Verteilte Transaktionen

Um über den gesamten Informationsbestand eines Unternehmens die Konsistenz zu sichern ist für Änderungen am Datenbestand, die für mehr als eine Anwendung gelten sollen, ein *2-Phase-Commit* das Mittel der Wahl. Dieses Transaktionsprotokoll sorgt für eine enge Kopplung der beteiligten Anwendungen: Sie müssen alle im Zeitraum der Transaktion verfügbar sein und in Bezug auf die Transaktion synchron arbeiten. Auch erbringen eigentlich die Anwendungen mit der Durchführung der Änderung den fachlichen Service gemeinsam.

Beispiele für verteilte Transaktionen finden sich in jeder Anwendungslandschaft, in der fachlich benachbarte, aber trotzdem getrennte Anwendungen auf getrennten Datenbeständen arbeiten. Auf der Ebene der Integration von Anwendungen gilt es, solche verteilten Transaktionen zu vermeiden. Wo der Architekt einer Umgebung aus Konsistenzgründen das Protokoll für verteilte Transaktionen einsetzen will, ist zu überprüfen, ob tatsächlich die Aufrechterhaltung der Konsistenz notwendig ist.

Transformationsmuster

Grundlage für den Einsatz von Transformationsmustern ist, dass die Anwendungen selbst wohl strukturiert sind, dass ihre Architektur Qualitätskriterien entspricht und dass sie sinnvoll modularisiert sind. Die Transformationsmuster sind hier allgemein formuliert, da die exakte Umsetzung von der technischen Infrastruktur und von den Entwicklungswerkzeugen der jeweils definierten SOA abhängt.

Kapselungsmuster

Kapselungsmuster zeichnen sich dadurch aus, dass die Transformation die eigentliche Anwendung nicht betrifft. Neue Softwarekomponenten zwischen der zu transformierenden Anwendung und der übrigen Anwendungslandschaft machen die fachlichen Funktionen als Service verfügbar.

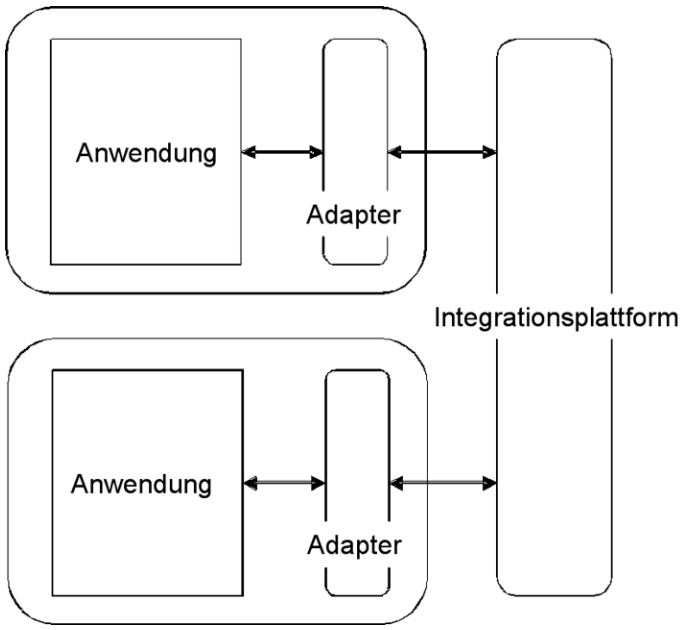

Abb. 1: Der Einsatz von Adaptern

Adapter

Eine zusätzliche Komponente, der Adapter, vermittelt Serviceanfragen und -antworten zwischen der Altanwendung und der übrigen Anwendungslandschaft entsprechend den festgelegten technischen Standards und im spezifizierten Kommunikationsmuster. Der Adapter nutzt in der Kommunikation mit der Anwendung definierte Schnittstellen oder reagiert auf bestimmte Ereignisse in der Anwendung. Mögliche Ansatzpunkte sind entfernten Prozeduraufrufe (RPCs) der Programmierschnittstelle der Anwendung und die Nutzung der Schnittstelle zwischen dem Anwendungskern und der Präsentationsschicht der Anwendung.

Adapter setzen auch Ereignisse der Anwendung in Servicekommunikation um. Ereignisse sind etwa die Erzeugung einer Datei oder die Veränderung eines Datenbankeintrags. Treten diese Ereignisse ein, kann dies eine Serviceanfrage zur Folge haben oder eine Serviceantwort auslösen. Ein Adapter verändert die Anwendung selbst nicht, er ist in diesem Sinn minimalinvasiv (siehe Abb. 1).

Einsatzgebiet von Adaptern ist die Überbrückung einer technischen Lücke zwischen der Anwendung und der technischen Plattform für die SOA. Sie sind also das Lösungsmuster für diejenige Klasse von Altanwendungen, die mit „Technische Lücke zwischen der Anwendung und der technischen Plattform für die SOA" überschrieben ist.

Die zusätzliche Software in den Kommunikationsbeziehungen kann den Durchsatz beeinträchtigen; deshalb ist das Zeit- und Lastverhalten von Adaptern genau zu prüfen und möglichst frühzeitig zu testen. Das gilt insbesondere für als Produkt bezogene Adapter, die sich einer weiteren Optimierung im Durchsatz verschließen.

Je nachdem, wo Adapter Ansatzpunkte in der Anwendung nutzen, verändert sich das nachgefragte Verhalten des Anwendungsnutzers. Ist etwa die Schnittstelle zwischen Anwendungskern und Präsentationsschicht der Ansatzpunkt, sind nicht das Zeitverhalten und das Mengengerüst in Bezug auf einen menschlichen Nutzer entscheidend für die Schnittstelle, sondern das auf andere Anwendungen. Tests müssen sicherstellen, dass die genutzten Ansatzpunkte diese Verhaltensänderungen verkraften beziehungsweise gewährleisten können.

Die Schnittstelle zwischen Adapter und Anwendung ist eng gekoppelt, d. h. Änderungen der Anwendung können immer auch Änderungen am Adapter zur Folge haben. Da der angebotene beziehungsweise

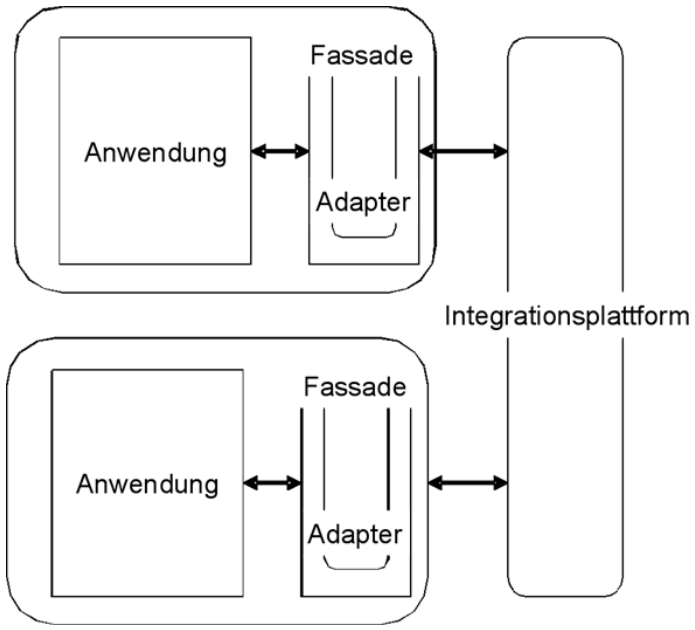

Abb. 2: Der Einsatz von Service-Fassaden

genutzte Service konstant sein soll, kann ein Service-Proxy schnell den Charakter einer Service-Fassade annehmen.

Service-Fassade

Die Service-Fassade ist eine Komponente, die Funktionen einer Anwendung so nach außen verfügbar macht, dass dort ein Service entsprechend den Qualitätskriterien nutzbar ist. Dafür muss die Service-Fassade verschiedene Ansatzpunkte in der Anwendung nutzen und koordiniert zusammenführen. Die Service-Fassade schließt die Funktion eines Adapters mit ein. Eine Service-Fassade setzt eine Serviceanforderung auf die koordinierte Nutzung mehrerer Schnittstellen der Anwendung um. Service-Fassaden sind in der Regel individuell entwickelt und sowohl von der individuellen Anwendung als auch von den angeforderten Services an dieser Stelle bestimmt (siehe Abb. 2).

Die Service-Fassade hat keine eigene persistente Datenhaltung. Sie wird aber, um zum Beispiel gegenüber einer Programmierschnittstelle eine Sitzung zu halten, bestimmte Informationen während eines Serviceaufrufs zwischenspeichern.

Einsatzgebiete sind Anwendungen des Typs „nur speziell definierte Schnittstellen, keine Services". Die Service-Fassade ist damit das Standardlösungsmuster für die überwiegende Menge der Altsysteme. Sie hat ihren Schwerpunkt bei der Nutzbarmachung der Funktionen von Individualanwendungen. Aber auch für den Aufbau standardgerechter Services auf der Grundlage einer Programmierschnittstelle eines Fertigprodukts setzt der Architekt eine Service-Fassade ein.

Die Service-Fassade kann nur dann auch unter Umständen viele Schnittstellen in der Altanwendung für das geforderte Serviceangebot zusammenführen, wenn die geforderte Funktion und Information zu dem Service in der Anwendung abstrakt vorhanden sind.

Hier gilt es – noch mehr als beim Einsatz von Adaptern – den Durchsatz-Gesichtspunkt und die Veränderung des Nutzungsverhaltens in der Anwendung zu überprüfen. Gerade die Zusammenfassung und Koordination von mehreren Schnittstellenaufrufen in der Anwendung kann ihre Leistung ernsthaft beeinträchtigen.

Die Definition neuer Schnittstellen und die Kapselung der Altanwendung können die ersten Schritte zu einer tief greifenden Veränderung und Migration der Anwendung selbst sein. Soll eine Anwendung sich zu einem der SOA besser entsprechenden Zielzustand verändern, ist der Aufbau einer Service-Fassade der erste Schritt (vgl. [Sch04]).

Nutzung eines fachlich allgemein definierten Services

Ein allgemein definierter und angebotener Service ersetzt die diesem gegenüber eingeschränkte fachliche Funktion einer Anwendung. Die Anwendung nutzt den allgemeingültigen Service, indem sie den anwendungsexternen Service statt der eigenen internen Funktion aufruft. An der Übergangsstelle bildet die Anwendung die gelieferten Informationen auf die benötigten ab. Die Abbildung zwischen allgemein gültigem Service und der – diesem gegenüber fachlich eingeschränkten – Anwendung findet in der nutzenden Anwendung statt (siehe Abb. 3).

Dieses Transformationsmuster hat seinen Anwendungsbereich bei der Klasse von „Anwendungen, die nur fachlich eingeschränkte Funktionen bieten". Es ist auch der erste Schritt in Richtung der Zerlegung einer Anwendung.

Die Umsetzung stellt einen tief greifenden und aufwändigen Eingriff in die Anwendung dar und setzt voraus, dass diese gut strukturiert ist und dass der Aufruf der internen Funktionen, die nun die Nutzung eines externen Services ersetzt, gut gekapselt ist.

Die Nutzung des allgemein gültigen Services erweitert nicht automatisch die Funktion einer fachlich eingeschränkten Anwendung. Es besteht ein Spannungs-

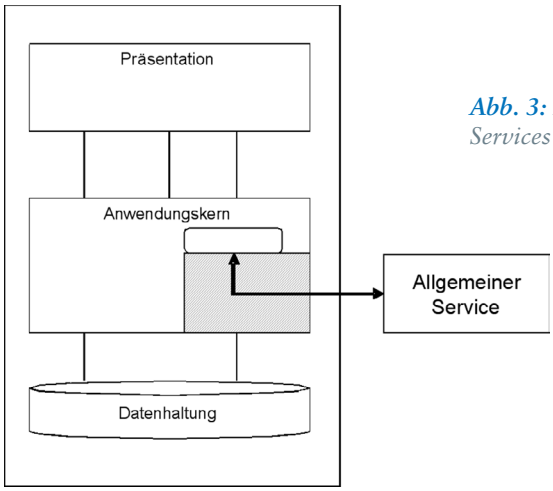

Abb. 3: Nutzung eines allgemeinen Services statt einer internen Funktion

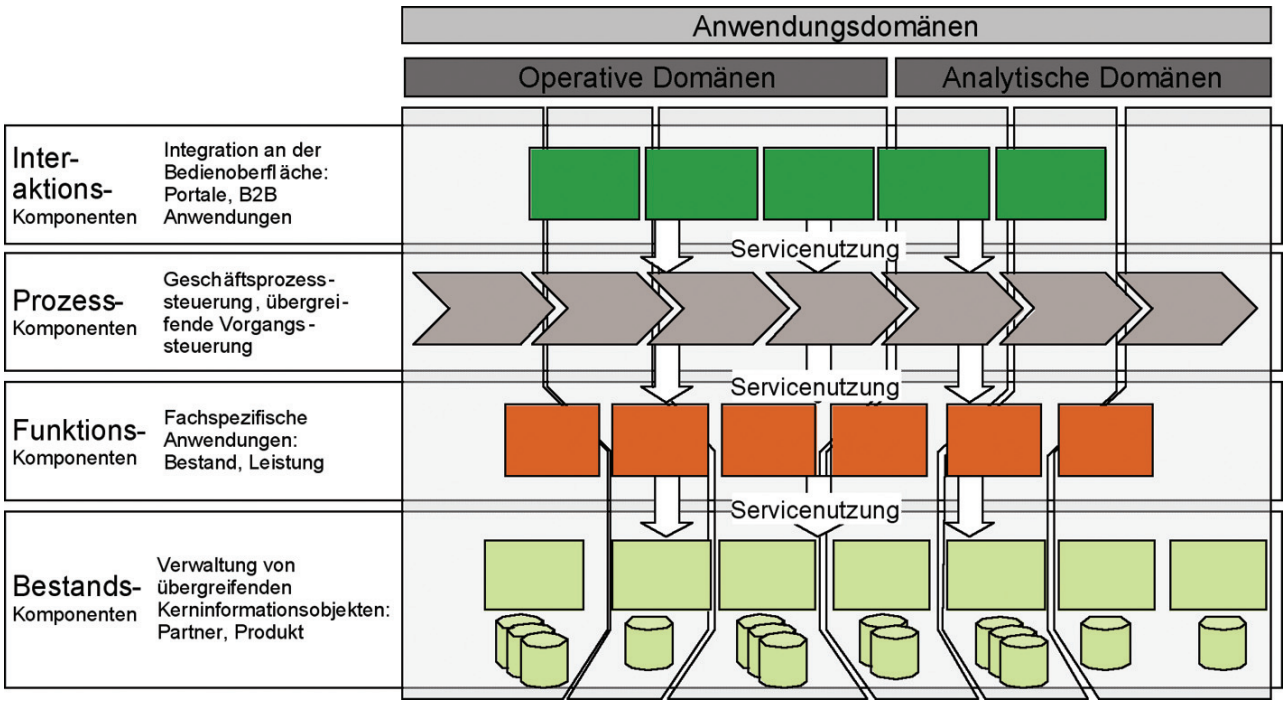

Abb. 4: Gliederung der Anwendungslandschaft nach Kategorien und Anwendungsdomänen

verhältnis zwischen dem von der Anwendung genutzten fachlichen Umfang und dem über den Service angebotenen.

Zerlegung

Möglichst kleine, standardisiert integrierbare Einheiten führen zu einer hohen Flexibilität in der Anwendungslandschaft. Kleine Bausteine, die schnell und einfach miteinander verknüpfbar sind, realisieren in ihrer Kombination auch komplexe und sich immer wieder verändernde Anforderungen schnell und einfach. Mehr Anwendungen verursachen aber auch einen höheren Entwicklungs- und Betriebsaufwand. Für Anwendungen existiert somit eine durch die Menge der angebotenen Services definierte optimale Größe. In bestehenden, realen Umgebungen sind Anwendungen häufiger zu groß als zu klein. Entsprechend bietet sich die Zerlegung von Anwendungen an (vgl. [Nan04]). Wo tatsächlich Anwendungen zu klein sind, sprechen die Kriterien, die eine Zerlegung nahe legen, auch für eine Zusammenführung.

Im Zuge der Umsetzung einer SOA ist es eine gute Vorgabe, Anwendungen unter Domänenaspekten in Kategorien zu zerlegen und Funktionen der technischen Infrastruktur von fachlichen Anwendungen zu separieren (vgl. [Hes06]). Interne Schnittstellen in einer aufzuteilenden Anwendung werden zu Services; Funktionen als technische Basiskomponenten sind zu doppeln oder besser als eigene Anwendungen bzw. Komponenten der technischen Infrastruktur mit entsprechenden Services auszuführen.

Zerlegung entlang der Domänen

Domänen strukturieren die fachliche Architektur einer Anwendungslandschaft. Sie sind dabei von einer bestimmten fachlichen Sicht durch Informationen und Regeln bestimmt. Der Grundsatz der *Separations of Concerns* gibt vor, dass Anwendungen nicht die Grenzen einer Domäne überschreiten sollen. So ist sichergestellt, dass sich Anwendungen und die von ihnen angebotenen Services auf einen bestimmten Zweck konzentrieren.

Zerlegung nach Anwendungskategorien

Reale Anwendungen fallen nach [Hes06] häufig in mehr als eine Anwendungskategorie. Unter dem Aspekt, Anwendungen kleiner und damit besser kombinierbar zu machen, kann auch eine Aufteilung nach den Kategorien „Interaktion", „Prozess", „Funktion" und „Bestand" sinnvoll sein. Diese Zerlegung nach Anwendungskategorien ist *keine* bloße Modularisierung einer Anwendung entlang ihren Schichten: Auch eine dann neue Anwendung der Kategorie „Bestand" (zum Beispiel eine Geschäftspartner-Verwaltung) hat eine Benutzungsoberfläche und setzt – wenn auch nicht besonders ausgeprägt – bestimmte algorithmische Funktionen um.

Damit ein SOA-Programm den versprochenen Nutzen erzielen kann, führt an der Zerlegung von Altanwendungen des Typs „Anwendungen mit zu vielen fachlichen Funktionen" kein Weg vorbei. Das Ergebnis sind dann aber wieder vollständige, wenn auch funktional fokussierte Anwendungen, die im Sinne von [Hes06] „sortenreine" Services anbieten. Abbildung 4 zeigt illustrativ eine so umgestaltete Anwendungslandschaft.

Ausgliederung von Funktionen der technischen Infrastruktur

Für große, umfassende Anwendungen gibt es die Tendenz zur eigenen technischen Infrastruktur. Deren Funktionen sind dann eng an den aktuellen Bedürfnissen und Besonderheiten dieser Anwendung ausgerichtet. Ziel ist es, mit der Definition der Infrastrukturkomponenten für die Anwendungslandschaft eine verbindliche und allgemeine Basis für die gesamte Umgebung zu schaffen. In der Folge ersetzen dann die allgemeinen Infrastrukturkomponenten die anwendungsspezifischen. Statt anwendungsinterne Schnittstellen zu nutzen, stützen sich alle Anwendungen in gleicher Weise auf die allgemeinen technischen Infrastrukturservices ab.

Die Ausgliederung von Funktionen der technischen Infrastruktur ist für Anwendungen vom Typ „Vermischung technischer Integration mit fachlichen Funktionen" geboten.

Muster in der Servicekommunikation

Auch bestimmte Muster in der Servicekommunikation unterstützen den Umbau zur SOA. Diese Muster passen die Art und Weise an, wie eine Anwendung über Services mit anderen Anwendungen kommuniziert. Sie unterstützen Zerlegungs- und Kapselungsmuster und helfen bei bestimmten Konstellationen von Altanwendungen.

Kompensierende Serviceoperationen

Die Vorgabe der losen Kopplung innerhalb einer SOA bedingt den Verzicht auf ein *2-Phase-Commit* in der verteilten Verarbeitung. Bleibt aber die Anforderung nach Transaktionen über mehrere Serviceaufrufe mit *ACID*-Eigenschaften (*Atomicity, Consistency, Isolation, Durability*) bestehen, sind kompensierende Serviceoperationen zu realisieren und einzusetzen, also Serviceoperationen, die komplementäre Wirkung zu einer anderen Serviceoperation entfalten. Eine Rückabwicklung einer fachlichen Transaktion erfolgt über den Aufruf der jeweils kompensierenden Serviceoperationen in der richtigen Reihenfolge. Scheitern in einer solchen Rückabwicklung die kompensierenden Services, wird allerdings eine manuelle Korrektur notwendig (siehe Abb. 5).

Für die Steuerung von Transaktionen gibt es einen Web-Service-Standard: *WS-BusinessActivity*. Wenn ein Service fachlich vollständig und normal ist, gibt es zu jeder Serviceoperation die kompensierende Serviceoperation (vgl. [Hes06]). Kompensierende Services sind damit das Lösungsmuster für die Umsetzung „Verteilter Transaktionen" in einer SOA.

Zwischenspeicherung

Eine Zwischenspeicherung ergänzt die Kapselungsmuster (siehe Abb. 6). Die Zwischenspeicherung verzögert die Zustellung von Serviceanforderungen an den Serviceanbieter. Die Anforderungen sind zunächst in einer Warteschlange, das heißt die Zwischenspeicherung hält jede ankommende Anfrage und leitet sie weiter, wenn die Anwendung in der Lage ist, die Serviceanfrage zu erfüllen – also zum Beispiel verfügbar ist oder als Batch-Anwendung läuft. Nach außen wirkt es so, als nähme eine mit Zwischenspeicherung gekapselte Anwendung Serviceanforderungen unmittelbar entgegen, tatsächlich verarbeitet sie diese zum passenden Zeitpunkt. Die Zwischenspeicherung ist so angelegt, dass die Speicherung im Prinzip für einen beliebig langen Zeitraum erfolgen kann.

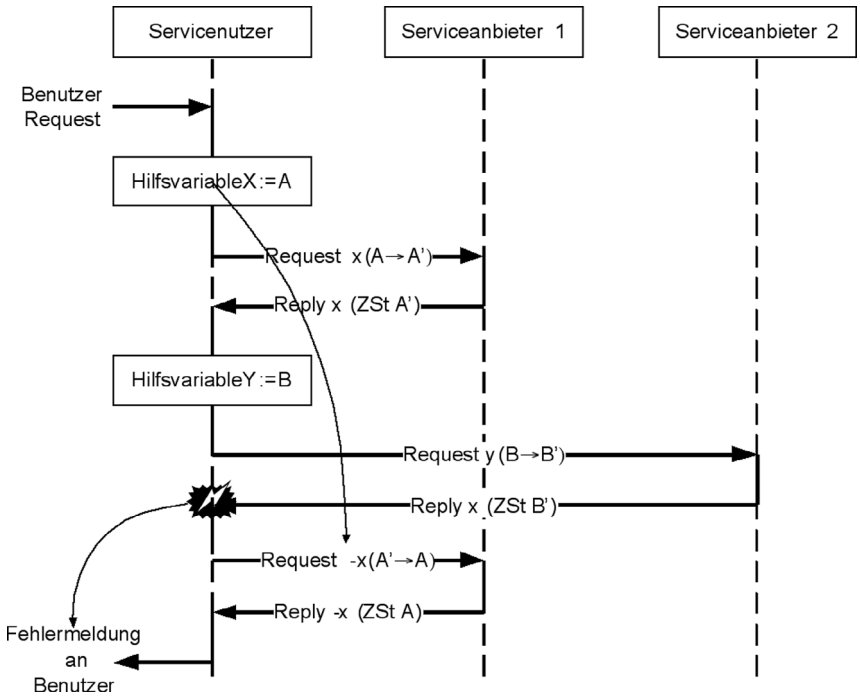

Abb. 5: Funktionsweise kompensierender Transaktionen

Eine Zwischenspeicherung ist notwendig für die Integration aller nicht dauernd verfügbaren Anwendungen in eine SOA. Sie ist also das Lösungsmuster für die Einbettung von *Batch-Ketten* in eine SOA.

Zwischenspeicherung ist auch ein geeignetes Mittel, um Serviceanforderungen zu priorisieren, in der Reihenfolge zu verändern und damit in eine gewünschte fachliche Reihenfolge zu bringen.

Das Vorgehen: Einsatz der Lösungsmuster

Grundlage jeder Konzeption für eine SOA ist die Sicht auf die gesamte Anwendungslandschaft. Die dargestellten Lösungsmuster und die Klassifikationen für Altanwendungen geben dem Architekten ein Werkzeug in die Hand, mit dessen Hilfe er den Umgang mit Altsystemen

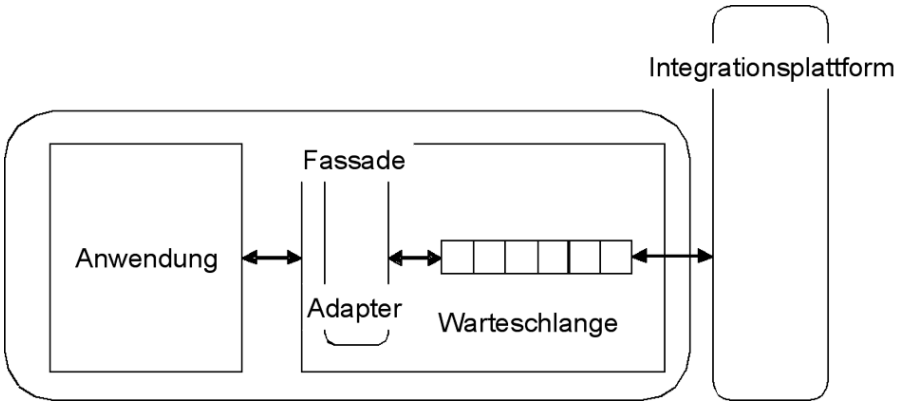

Abb. 6: Funktionsweise Zwischenspeicherung für nicht dauernd verfügbare Anwendungen

in Form von Richtlinien in einem Programm zur Umsetzung einer SOA festlegen kann. Dazu konkretisiert er zunächst die Lösungsmuster im Hinblick auf die technische Basis und definiert dann ihren Einsatz, orientiert an den Klassifikationen der tatsächlich vorgefundenen Anwendungen.

Das Programm für die Einführung einer SOA hat in der so beschriebenen Entwurfsphase die gesamte Anwendungslandschaft im Blick; deshalb sind die Richtlinien auf der Basis von Mustern übergreifend formuliert. Ein einzelner Service oder die Umstellung der Kommunikation zwischen zwei Anwendungen kann nur ein Pilot sein und niemals die notwendige Gesamtkonzeption ersetzen.

Genauso notwendig wie diese geschlossene Gesamtkonzeption sind die Mitwirkung der Fachseite und die Orientierung an den Geschäftsprozessen des Unternehmens. Insbesondere für die Zerlegung von Anwendungen, die Realisierung fachlich allgemein definierter Services und für die Beschreibung kompensierender Serviceoperationen ist die Zusammenarbeit mit der Fachseite zwingend notwendig. Mit der Fachseite ist auch abzustimmen, welche Einschränkungen das Unternehmen in der langen Umbauphase zu einer SOA hinnehmen kann. Der Architekt nutzt die Klassifikation und die Lösungsmuster als Ausgangspunkt für die Planung zur Umgestaltung der Anwendungslandschaft. Den endgültigen Plan muss er aber zusammen mit allen Betroffenen und Interessengruppen anhand der Anforderungen konkret aufstellen. ■

Literatur & Links

[Erl04] T. Erl, Service-oriented Architecture. A Field Guide to Integrating XML and Web Services, 2004 Prentice Hall International

[Hes04] A. Hess, Sie sind es uns wert!, in: IT-Management 8/04

[Hes06] A. Hess, B. Humm, M. Voß, Regeln für service-orientierte Architekturen hoher Qualität, in: Informatik Spektrum 6/06

[Nan04] O.F. Nandico, Von Monolithen zu lose gekoppelten Services, in: IT-Fokus 9-10/04

[Pet06] C. Pettey, L. Goasduff, Gartner's 2006 Emerging Technologies Hype Cycle Highlights Key Technology Themes, in: Gartner Press Releases 2006, siehe: www.gartner.com/it/page.jsp?id=495475

[Sch04] P. Schaumann, Altsysteme von außen nach innen ablösen, in: IT-Fokus 7-8/04

[Sie03] J. Siederleben (Hrsg.), Softwaretechnik, Carl Hanser Verlag 2003