

mehr zum thema:

www.spmtechnologies.com/de/customers/case_studies/sbb.html
www.service-architecture.com

SOA – WERTSTIFTENDES ARCHITEKTUR-PARADIGMA

Das Schlagwort SOA (Service Oriented Architecture) wird in letzter Zeit immer wieder im Zusammenhang mit Web-Services genannt. SOA ist jedoch ein von Web-Services unabhängiges Architektur-Paradigma, das auf Basis verschiedener Technologien realisiert werden kann. Der Artikel entwickelt, ausgehend von einem konkreten Geschäftsfall, die wesentlichen Elemente einer serviceorientierten Architektur. Hintergrund der Betrachtungen sind mehrjährige Erfahrungen bei der Umsetzung einer serviceorientierten Architektur in einem deutschen Logistikunternehmen.

Die Zeiten, in denen jede halbwegs Heil versprechende Technologie zu einem Hype und einer Reihe damit verbundener kostspieliger Großprojekte geführt hat, sind vorbei. Bei den CIOs dieser Welt ist die eigentliche Wertschöpfung eines IT-Systems das Maß aller Dinge geworden. Vielleicht kann dieser durch die Krise der letzten Jahre herbeigeführte Zustand auch als eine mögliche Erklärung dafür dienen, dass viel versprechende Technologien wie beispielsweise Web-Services bisher über die Prototypenphase nicht wirklich hinausgekommen sind. Jeder, der schon einmal versucht hat für einen IT-Verantwortlichen eine Wirtschaftlichkeitsberechnung für den Einsatz von Web-Services im Unternehmen zu erarbeiten, kann davon ein Lied singen.

Mit SOA hat sich jedoch in jüngster Zeit ein Begriff seinen Weg in die Analystenberichte gekämpft, dessen Mehrwert für große Organisationen leicht ersichtlich und nachweisbar ist. Inzwischen ist unumstritten, dass es sich bei dem Konzept „Service-Orientierung“ nicht um ein weiteres Strohfeuer zur Generierung eines Scheingeschäfts handelt. Vielmehr sprechen mittlerweile mehr als nur eine Handvoll Technologie-Verliebte gar von der nächsten großen IT-Revolution.

Fakt ist jedenfalls, dass die großen Hersteller von Integrationssoftware (IBM, Microsoft, SAP, BEA) alles daran setzen SOA für die eigenen Produkte („WebSphere“, „.NET“, „NetWeaver“, „WebLogic“) in Anspruch zu nehmen.

Der Grund für die guten Erfolgchancen von SOA ist wohl vor allem, dass es sich hierbei um ein IT-Konzept handelt, bei dem die eigentlichen Geschäftsprozesse im Vordergrund stehen. Die realisierende Technologie tritt hingegen mehr und mehr

in den Hintergrund. Endlich scheint die Erfüllung des Traums einer jeden Fachabteilung näher zu rücken: Die IT wird wirklich dazu verwendet, die abzuwickelnden Geschäftsvorfälle zu unterstützen und nicht umgekehrt. Angenehmer Nebeneffekt – und nicht mehr Mittel zum Zweck – soll nach erfolgreicher IT-gestützter Umsetzung dann die Definition und schnelle Realisierung neuer, optimierter Geschäftsprozesse sein, die einen echten Wettbewerbsvorteil bedeuten.

Zudem soll SOA dazu verhelfen, das so genannte Echtzeit-Unternehmen Realität werden zu lassen. Ein solches hat zu jeder Zeit einen vollständigen Überblick über alle geschäftsrelevanten Daten und zugleich die Möglichkeit eines frühzeitigen und gezielten Reagierens bei unerwarteten Marktveränderungen.

Grund genug, einmal einen genaueren Blick auf eine konkrete Realisierung einer SOA in einem realen Projekt zu werfen. Dieser Artikel stellt dabei vor allem die Besonderheiten der Vorgehensweise bei der Erarbeitung von SOA in den Vordergrund. Konkrete Technologien und Produkte spielen hierbei eine Nebenrolle.

Das Beispielszenario

Ein Logistik-Unternehmen möchte ein professionell arbeitendes Callcenter zur kurzfristigen und kompetenten Bearbeitung von Kundenreklamationen aufsetzen, um die Kundenbetreuung zu verbessern und eine engere Kundenbindung herzustellen. Im Unternehmen gibt es bereits mehrere umfangreiche Kundendatenbanken, die aus historischen Gründen nebeneinander existieren.

Das Callcenter wird durch ein beauftragtes Unternehmen eingerichtet und mit Personal

► die autoren



Dr. Bärbel Burkhard (E-Mail: barbara.burkhard@spmtechnologies.com) ist Group Manager im Bereich Softwareentwicklung bei SPM Technologies und arbeitet zur Zeit in einem Kundenprojekt an einer unternehmensweiten Integrationslösung auf der Basis service-orientierter Architekturen.



Guido Laures (E-Mail: guido.laures@spmtechnologies.com) verantwortet als Vice President bei SPM die Softwareentwicklung. Schwerpunktmäßig beschäftigt er sich im Rahmen von Integrationsprojekten mit Konzepten, Methoden und Technologien zur Optimierung der Wertschöpfung IT-gestützter Geschäftsprozesse.

ausgestattet. Die Callcenter-Mitarbeiter sollen Beschwerden telefonisch entgegennehmen und zur Bearbeitung weiterleiten. Sie benötigen dazu keine genaue Fachkenntnis über den Gegenstand der Reklamation. Die Reklamationsbearbeitung selbst erfolgt durch die jeweils spezialisierten Mitarbeiter des Logistikunternehmens. Dabei soll ein bereits vorhandenes Ticketing-System weiterhin eingesetzt werden. Der Kunde wird nach Abschluss der Reklamationsbearbeitung bzw. bei lang andauernder Bearbeitungszeit über Zwischenstände informiert. Dabei ist das Prinzip „single face to the customer“ einzuhalten, d. h. wo ein persönlicher Kontakt zum Kunden angemessen ist, soll dies immer über denselben Callcenter-Mitarbeiter geschehen. Informationen zum abgeschlossenen Reklamationsvorgang sollen in eine vorhandene unternehmensweite *Business-Intelligence*-Komponente eingepflegt werden, um Auswertungen von Reklama-

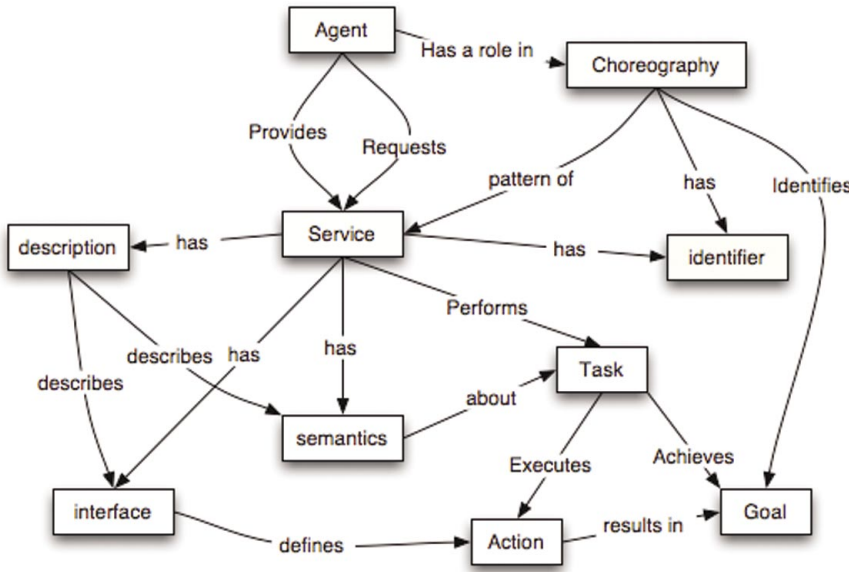


Abb. 1: Das beim W3C in Arbeit befindliche Service Oriented Model (aus: [Boo03])

tionsgründen, Bearbeitungszeiten etc. zu ermöglichen.

Ableitung einer SOA

Im Folgenden wird in einem *Top-Down*-Ansatz aus dem beschriebenen Geschäfts-szenario heraus abgeleitet, wie Elemente einer serviceorientierten Architektur identifiziert und Gewinn bringend eingesetzt werden können. Wir benutzen die Terminologie des *Service Oriented Model (SOM)* (siehe Abb. 1), das als Bestandteil der *Web Services Architecture (WSA)* derzeit vom W3C erarbeitet wird (vgl. [Boo03]).¹⁾ Die WSA ist eine web-service-spezifische Ausprägung einer serviceorientierten Architektur. Sie versteht sich als Rahmen für die Weiterentwicklung künfti-

ger web-service-bezogener Standards. Dieser Rahmen wird gebildet durch fünf Modelle, die jeweils unterschiedliche Aspekte der Web-Service-Architektur beschreiben:

- *Service Oriented Model (SOM)*
- *Resource Oriented Model (ROM)*
- *Message Oriented Model (MOM)*
- *Policy Model (PM)*
- *Management Model (MM)*

Die Ableitung einer serviceorientierten Architektur bewegt sich im Spannungsfeld zwischen Geschäftsprozess, Service-Modell und plattformspezifischer Umsetzung (siehe Abb. 2). Business-Analyst, Service-Architekt und Entwickler haben unterschiedliche Sichten auf eine SOA, die sich gegenseitig beeinflussen.

¹⁾ Hier genutzte Definitionen finden sich in Kasten 1)

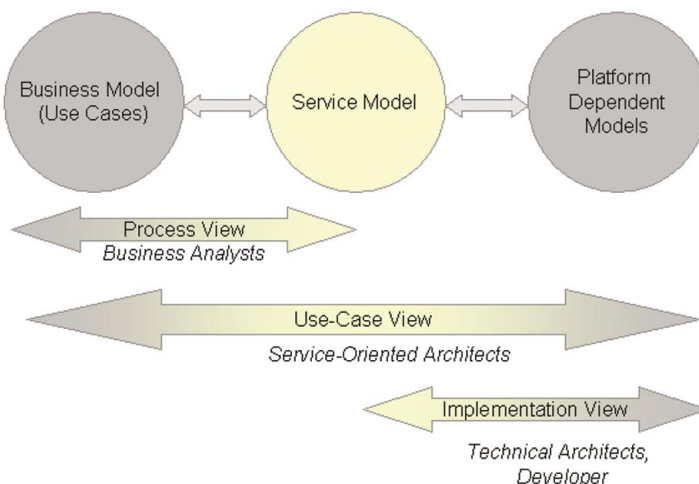


Abb. 2: Vorgehensmodell bei der Entwicklung einer SOA (Quelle: ZapThink LLC, in [Blo03])

Das Business-Modell: Beschreibung des Geschäftsteilprozesses

Ausgangspunkt ist die Arbeit des Business-Analysten. Basierend auf den vorliegenden Anforderungen modelliert er den Geschäftsteilprozess „Reklamationsbearbeitung“. Die Modellierung eines Geschäftsteilprozesses beschreibt, welche Prozessschritte in Abhängigkeit von eintretenden internen und externen Ereignissen zu durchlaufen sind und welche gültigen Endzustände zu erreichen sind – dies alles aus einer geschäftsprozess-zentrierten Sicht (*Process View*). Hierbei geht es vor allem auch darum, eine vollständige *Stakeholder*-Analyse der Organisation durchzuführen. Die Fallstricke bei der Realisierung einer SOA liegen oft nicht in der Technik begründet, sondern in den Unklarheiten der Verantwortlichkeitsbereiche einzelner Fachabteilungen. Typischerweise werden zur Modellierung Use-Cases in UML-Notation (vgl. [Fow99]) verwendet.

Die Use-Case-Diagramme modellieren die statische Anwendungssicht (siehe Abb. 3), die durch die entsprechende Geschäftsprozessmodellierung unterlegt wird.

Entscheidende Ergebnisse dieser Modellierung sind die Identifikation von Teilaufgaben (*Tasks*, siehe Kasten 1). Für unser Beispielszenario haben wir mit der Modellierung des Geschäftsteilprozesses „Reklamationsbearbeitung initialisieren“ die *Tasks*

- „Kunden suchen“,
- „Reklamation anlegen“,
- „Reklamationsbearbeitung initialisieren“ und
- „Kundenkontakt festhalten“

identifiziert (siehe Abb. 4).

Beim Geschäftsteilprozess „Reklamationsabschluss bearbeiten“ (siehe Abb. 5) sind es die *Tasks*

- „Reklamation bearbeiten“
 - „Reklamation abschließen“
 - „Kundenvorgang festhalten“ und
 - „Kunde über Abschluss informieren“.
- Diese Beispielmodellierung dient lediglich der Veranschaulichung und ist bei Weitem nicht vollständig.

Das Service-Modell: Design von Business-Services

Nachdem die fachlichen Abläufe der Geschäftsvorfälle spezifiziert sind, ▶

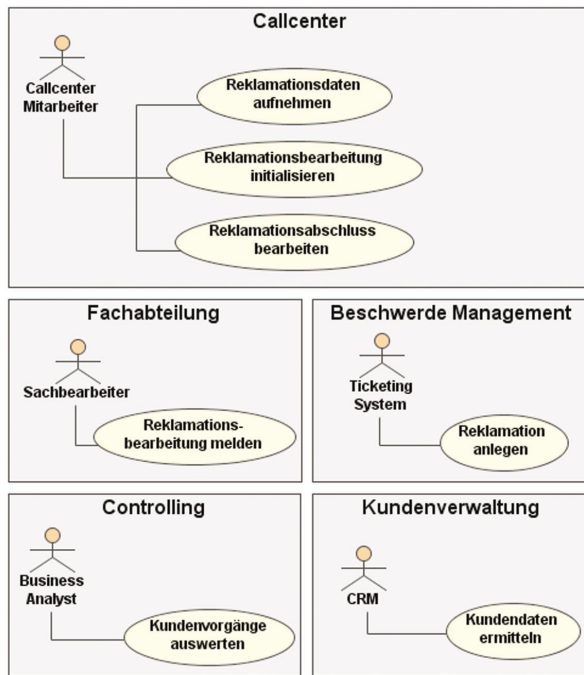


Abb. 3: Stakeholder und Use-Cases der Reklamationsbearbeitung

kommt der Service-Architekt verstärkt zum Zuge. Er muss aus dem erarbeiteten Modell *Services* ableiten und festlegen, in welcher Abfolge unter Beachtung welcher Abhängigkeiten dieses genutzt werden können, um die gewünschte Funktionalität zu erbringen (*Choreography*). Ein Service sollte eine oder mehrere *Tasks* umsetzen.

Hierbei ist die Wahl einer geeigneten Granularität entscheidend: Fein-granulare Servicedefinitionen führen zu generischen Softwarebausteinen mit geringem Funktionsumfang, deren Wiederverwendbarkeit hoch ist. Andererseits sinkt ihre tatsächliche Verteilbarkeit, da bei Prozessen, die durch den Aufruf vieler, sehr klein

geschnittener Services umgesetzt werden, der zusätzliche Kommunikationsaufwand – insbesondere der Einfluss netzbedingter Verzögerungen – sehr hoch werden kann.

Grob-granulare Services sind für die Verteilung besser geeignet; außerdem wird die Choreographie zur Umsetzung des Prozesses einfacher. Allerdings kann der Grad der Wiederverwendbarkeit bei grob-granularen Services sinken. Hier liegt die besondere Verantwortung des Service-Architekten. Er muss *Service-Interface* und *Service-Semantik* so bestimmen, dass sie von möglichst vielen potenziellen Servicenutzern (*Service Requester*) akzeptiert werden können. Dazu muss der Service-Architekt einen Überblick über die zu realisierenden Geschäftsprozesse sowie die vorhandene und avisierte Anwendungslandschaft des Unternehmens – den so genannten Bebauungsplan – haben. Er muss das *Servicedesign*, d.h. die Aushandlung der Semantik und der Schnittstellen von Services, als einen kooperativen Prozess unter Einbindung aller Beteiligten, insbesondere der Business-Analysten, begreifen, den er federführend vorantreibt. *Servicedesign* ohne Kenntnis des Nutzungskontextes für einen anonymen Service-Markt ist aus der allgemeinen Projekterfahrung heraus kein geeignetes Vorgehen für die unternehmensweite Business-Integration.

Action	Die Aktivität, die ein <i>Agent</i> als Reaktion auf den Erhalt einer Nachricht ausführt oder die dazu führt, dass er eine Nachricht sendet.
Agent	Ein Programm, das im Auftrag einer Person, einer Organisation oder eines Prozesses agiert. Es kann <i>Services</i> anbieten und aufrufen.
Choreography	Die Reihenfolge und Bedingungen, unter denen mehrere voneinander unabhängige (Web-)Services agieren, um eine bestimmte Funktionalität zu erbringen.
Choreography Description Language	Eine formale, maschinell verarbeitbare Beschreibungssprache für Choreographien. Sie legt die erlaubten Interaktionen zwischen einer Menge von <i>Services</i> fest und kann den Lebenszyklus eines Aufrufs sowie mögliche Konversationen zwischen <i>Service Requester</i> und <i>Service Provider</i> beschreiben.
Service	Ein <i>Service</i> ist eine Sammlung zusammengehöriger <i>Tasks</i> . Sie werden durch einen oder mehrere <i>Agents</i> realisiert, die als <i>Service Provider</i> agieren. Ein <i>Service</i> ist charakterisiert durch: <ul style="list-style-type: none"> ■ einen eindeutigen Identifikator ■ ein <i>Service Interface</i> ■ bestimmte <i>Service Semantics</i> ■ Er sollte dynamisch publizierbar und auffindbar sein sowie den entfernten Zugriff über einen <i>Service End Point</i> erlauben.
Service End Point	Eine Netzwerk-Adresse, an der eine Implementierung des <i>Service Interface</i> bereitgestellt wird.
Service Interface	Abstrakte Beschreibung des <i>Services</i> als Menge von Service-Operationen. Über ein Interface wird definiert, welche Nachrichtenfolgen der <i>Service</i> empfangen und senden kann.
Service Provider	Ein <i>Agent</i> , der die <i>Actions</i> eines oder mehrerer Services ausführen kann bzw. ihre Ausführung auslöst.
Service Requester	Ein <i>Agent</i> , der einen Service aufruft.
Service Semantics	Der Vertrag zwischen <i>Service Requester</i> und <i>Service Provider</i> , die Voraussetzungen für und die Auswirkungen der Servicenutzung betreffend.
Task	Eine <i>Task</i> hat ein Ziel und umfasst eine oder mehrere <i>Actions</i> , die zu diesem Ziel führen. <i>Tasks</i> sind <i>Services</i> zugeordnet.

Kasten 1: Begriffsdefinition des SOM (vgl. [Boo03])

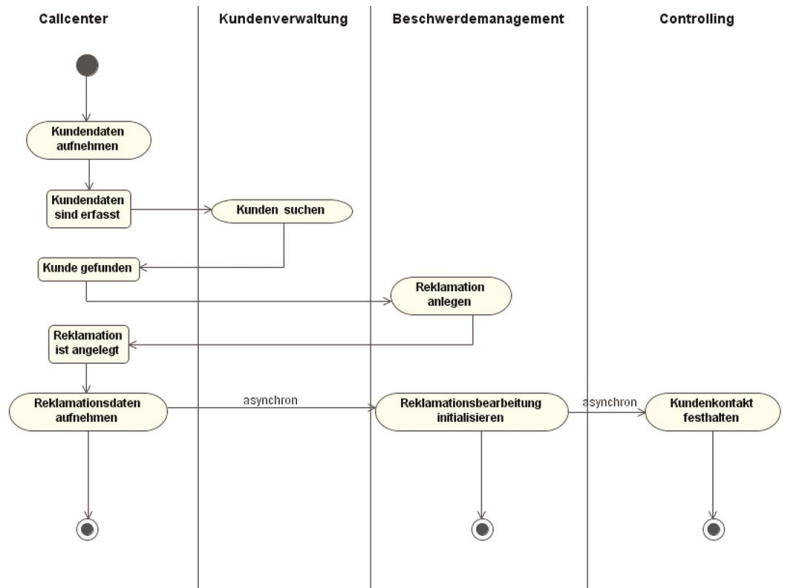


Abb. 4: Geschäftsprozessmodell „Reklamationsbearbeitung initialisieren“

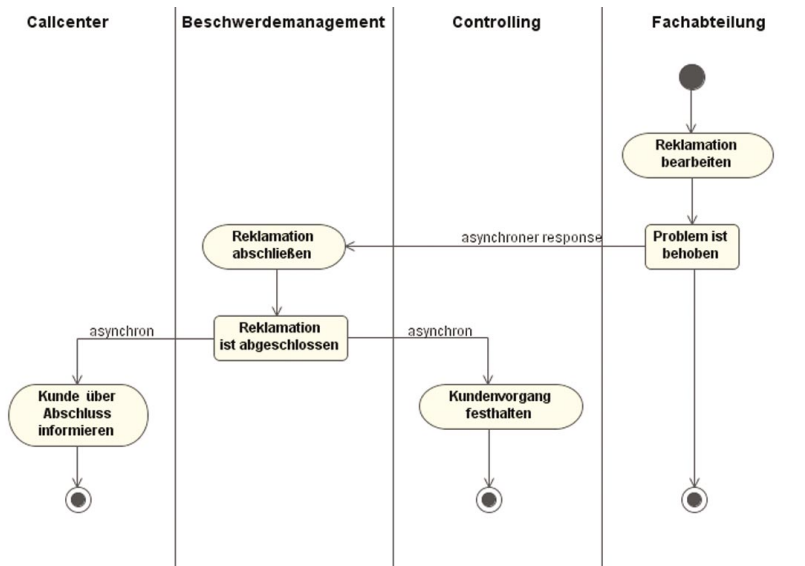


Abb. 5: Geschäftsprozessmodell „Reklamationsabschluss bearbeiten“

Als Bestandteil der Servicedefinition wird festgelegt, wie der zeitliche Ablauf der Interaktionen zwischen *Service Requester* und *Service Provider* ist. Wartet der *Service Requester* auf die Beendigung des Serviceaufrufs (synchroner *Request*) oder arbeitet er nach Aufruf des Services weiter und lässt sich das Ergebnis der Serviceausführung asynchron zustellen? Denkbar ist auch, dass er nicht an einer Rückmeldung des Service interessiert ist, sondern dass ihm die zuverlässige Zustellung des *Request* ausreicht (*oneway*). Schließlich könnte sich der *Requester* für

den Empfang einer ganzen Klasse von Nachrichten registrieren (*publish/subscribe*). Welche dieser Interaktionsstile tatsächlich unterstützt werden und ob es eventuell noch andere gibt, hängt von der Realisierung der SOA, also dem plattformspezifischen Modell, ab.

Die oben genannten Interaktionsstile „synchron“, „asynchron“, „oneway“ und „pubsub“ werden durch eine speziell für diesen Kunden entwickelte SOA-Plattform unterstützt, die seit Jahren bei ihm produktiv ist.

Aus dem modellierten Geschäftsprozess

kann man mehrere Services ableiten:

CustomerManagement

Der Service CustomerManagement (siehe [Tabelle 1](#)) erlaubt das Auffinden von Kundendaten. Das Anlegen, Modifizieren und Löschen von Kundendaten sind für den vorliegenden Geschäftsfall nicht relevant. Um den Service in anderen Use-Cases besser wiederverwenden zu können, würde ein Service-Architekt diese Methoden wahrscheinlich ergänzen. Die Schnittstelle (*Interface*) des Service CustomerManagement bietet bei erstem Hinsehen kaum mehr als einen entfernten Datenbankzugriff. Die Kapselung als Service ist trotzdem sinnvoll, weil die damit eingeführte Abstraktionsschicht die in unserem Beispiel verteilte Kundendatenhaltung für den *Service Requester* transparent macht. Soll beispielsweise die Kundendatenverwaltung um einen Prozess ergänzt werden, in dem mögliche „Dubletten“ (Einträge, die zwar unterschiedliche Angaben zu Firmennamen, Adresse etc. enthalten, sich aber auf denselben Kunden beziehen) erkannt werden, kann dies unter Beibehaltung der Serviceschnittstelle und ohne Auswirkungen auf den *Service Requester* erfolgen.

ComplaintProcessing

Der Service ComplaintProcessing (vgl. [Tabelle 2](#)) bietet eine Methode zum Anlegen des Reklamationsvorgangs, deren Ergebnis (eine Reklamations-ID) beispielsweise dem anrufenden Kunden als Referenz auf die Reklamation ausgehändigt werden kann. Mit Aufruf der Methode `processComplaint` wird die Steuerung des Workflows vom *Agenten* des Callcenter-Mitarbeiters an den *Agenten*, der den Service ComplaintProcessing implementiert, übergeben.

Zu entscheiden ist hier, ob die Interaktionen, welche die einzelnen Sachbearbeiter gegenüber dem Beschwerdemanagementsystem ausführen, wie z.B. „Beginne die Bearbeitung der Reklamation X“, „Ändere Status der Reklamation Y“, „Weitergabe des Reklamationsvorgangs an Abteilung 0123“ oder „Bearbeitung der Reklamation Z beendet“, auch auf Serviceaufrufe abgebildet werden sollen. In dem hier vorgestellten Projekt fiel die Entscheidung dagegen. Das

Methoden	Beschreibung	Realisierte Tasks	Interaktionsstil
findCustomer	Liefert alle Kundendatensätze zu angegebenen Suchkriterien	Kundendaten ermitteln	synchron

Tabelle 1: Service CustomerManagement



Service-Interface bietet nur Methoden, die außerhalb der Fachabteilungen benötigt werden. Die Sachbearbeiter nutzen weiterhin die bekannte Oberfläche des Ticketing-Systems.

Ein *Service Requester* kann sich für den Erhalt von Änderungsmeldungen bezüglich der Zustandsänderung einer Reklamation registrieren. Die Servicemethode `newComplaintStates` stellt solche Nachrichten auf dem zugeordneten *Topic* bereit. Der *Service Requester* kann die erhaltenen Meldungen lokal verwalten. Dadurch kann eine Kundennachfrage sofort, ohne Aufruf einer weiteren Servicemethode, beantwortet werden. Falls der Aufwand für die Verwaltung von Statusmeldungen beim *Requester* zu hoch erscheint, könnte der Service um eine synchrone Lesemethode `getStatus` ergänzt werden, mit dem Nachteil, dass eventuell auftretende Verzögerungen auch für den Anrufer spürbar werden.

Zur Aufnahme bearbeiteter Reklamationsvorgänge in die Business-Intelligence-Komponente kann sich ein weiterer Agent für den Erhalt der Nachrichten von `newComplaintStates` registrieren. Er wird nur Meldungen, die den Abschluss eines Reklamationsvorganges mitteilen, verarbeiten und alle anderen vom *Topic* zugestellten Nachrichten verwerfen. Das *Data Warehouse* tritt hier also nur als *Service Requester* auf. In anderen Use-Cases wird es sicher angebracht sein, eine Anbindung als *Service Provider* zu realisieren und Methoden z. B. zur Erstellung von Analysen über dem Datenbestand zu modellieren. **Abbildung 6** illustriert das Beispielszenario.

Service-Definition

Für die Nutzung eines Services ist es ausreichend, Interface, Semantik und Identifi-

kator zu kennen. Wie und wo der Service implementiert ist, sollte für seine Nutzung unerheblich sein. Dadurch ist die Austauschbarkeit der Serviceimplementierungen bei stabilem Interface und stabiler Semantik möglich.

Die bisher vorliegende verbale Servicebeschreibung ist nicht annähernd ausreichend als Grundlage für eine Service-Implementierung. An dieser Stelle wird deutlich, dass der Service-Architekt Anforderungen aus dem plattform-spezifischen Modell einfließen lässt: Er muss sich für eine bestimmte Schnittstellenbeschreibungssprache entscheiden. Da der Servicegedanke seit Langem in unterschiedlichen Kontexten verwendet wird, gibt es entsprechend unterschiedliche Möglichkeiten zur Beschreibung der Serviceschnittstelle. DCOM und CORBA unterstützen die komponentenbasierte Programmierung und damit ein eher objektorientiertes als ein serviceorientiertes Herangehen. Interfaces sind verstärkt datenzentriert. Häufig ist zu beobachten, dass das Interface ein „Ding“ bezeichnet, mit dem der Aufrufer Verschiedenes tun kann, es z. B. erzeugen, Zustände erfragen, modifizieren oder löschen. Damit wird die Möglichkeit geboten, entfernt auf das Businessobjekt zuzugreifen, was der Denkweise objektorientierter Entwicklung nahe kommt. Werden generierte schnittstellenspezifische *Stubs* und *Skeletons* eingesetzt, führt dies zu einer engen Kopplung des Client-Codes an ein bestimmtes Interface.

Die Beschreibung der Service-Schnittstelle mit XML, wie z. B. in Web-Services, bietet dem gegenüber Vorteile. Zum einen ist XML mittlerweile als Beschreibungssprache für viele Belange akzeptiert und auch außerhalb von Softwareentwicklungskreisen im Einsatz. Aus der Prozesssicht

(*Process View*, siehe **Abb. 2**) heraus in XML modellierte *Service Interfaces* beschreiben häufiger einen Vorgang, den man auslösen kann und über dessen Bearbeitungszustände man sich informieren kann. Das eigentlich bearbeitete Businessobjekt muss nicht am *Service Interface* sichtbar sein, sondern nur seine für den jeweiligen Service relevanten Aspekte.

Ein weiterer entscheidender Vorteil von Web-Services ist die Allgegenwart (*Ubiquity*) ihrer Grundlagen: Mit XML und HTTP nutzen Web-Services Standards, die inzwischen überall ohne jegliche Zusatzinstallationen verfügbar sind. In CORBA oder DCOM angebotene Services setzen dem gegenüber beim *Service Requester* voraus, dass er ebenfalls dieses Programmiermodell unterstützt, was im Allgemeinen nicht von vornherein der Fall ist. Mit DCOM ist man damit bezüglich der Plattformen eingeschränkt, aber auch bei CORBA hat sich gezeigt, dass Interoperabilitätsprobleme auftreten können, wenn ORBs unterschiedlicher Hersteller eingesetzt werden.

Die Reihenfolge und die Bedingungen, unter denen die entworfenen Services agieren, um gemeinsam den Teilgeschäftsprozess „Reklamationsbearbeitung“ zu erbringen, werden in der *Choreography* beschrieben. Idealerweise sollte diese formal spezifiziert vorliegen und ausführbar sein. Für Web-Services etabliert sich die *Business Process Execution Language (BPEL/BPEL4WS)* (vgl. [BPE03]) als Beschreibungssprache. Der Beitrag „Choreographie: Geschäftsprozesse mit Web-Services“ von Frank Leymann auf S. 13 in diesem Heft beschäftigt sich genauer mit diesem Thema.

Methoden	Beschreibung	Realisierte Tasks	Interaktionsstile
<code>createComplaint</code>	Eine initial erfasste Reklamation wird auf Vollständigkeit und Konsistenz geprüft und in der Reklamationsverwaltung aufgenommen. Die gelieferte Reklamations-ID dient als eindeutiger Bezug auf den Reklamationsvorgang.	Reklamationsdaten aufnehmen	synchron
<code>processComplaint</code>	Eine vollständig erfasste Reklamation wird bearbeitet. Die Bearbeitung durchläuft mehrere Zustände. Über jede Zustandsänderung kann sich der Aufrufer informieren lassen. Das endgültige Ergebnis der Reklamationsbearbeitung wird als Response bereitgestellt.	Reklamationsbearbeitung initialisieren Interne Nachbereitung des Reklamationsfalls (Aufruf von <code>enterContact</code>)	asynchron
<code>newComplaintStates</code>	Für jede Zustandsänderung in der Bearbeitung einer Reklamation wird eine Nachricht unter Angabe der Reklamations-ID versendet.	Behandeln einer Statusanfrage	pubsub

Tabelle 2: Service ComplaintProcessing

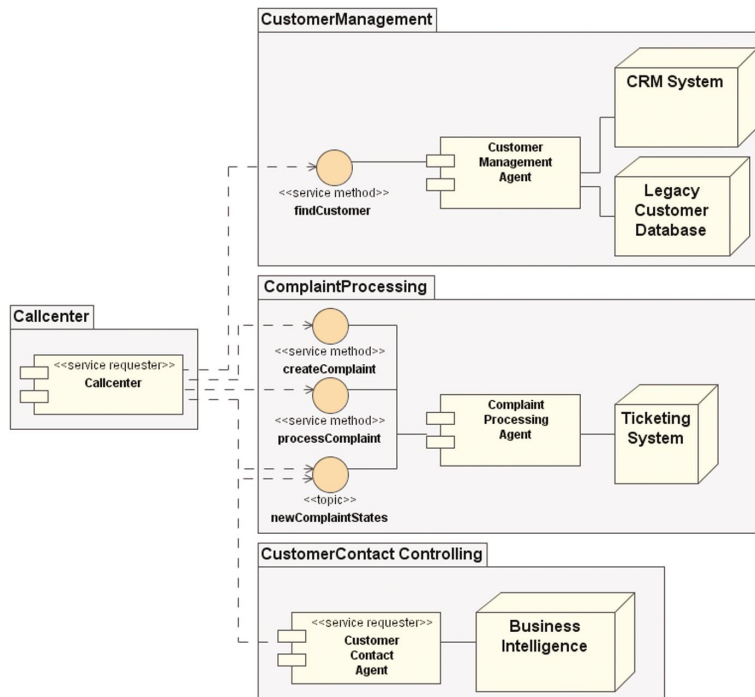


Abb. 6: Service Requester und Service Provider

Serviceaggregation in Domänen

Serviceorientierte Architekturen bergen die Gefahr, dass Services (re-)definiert werden und bereits vorhandene Funktionalität über ein verändertes Interface anbieten. Die Ursache dafür ist, dass ein Service jeweils für einen speziellen Use-Case maßgeschneidert wird. Je umfangreicher die abzudeckenden Geschäftsprozesse sind, um so schwieriger wird es für den Service-Architekten, solche Fehlentwicklungen frühzeitig zu erkennen und zu verhindern. Hilfe kann die Einführung von Domänen bieten. Eine Domäne ist eine logische Zusammenfassung von Services. Dabei kann man zwischen Inter-Domänen- und Intra-Domänen-Aspekten unterscheiden.

Inter-Domänen-Aspekte beschreiben die externe Sicht auf Domänen. Eine Domäne fasst Services zusammen, die unterschiedliche Aspekte eines Businessobjekts abdecken. Angenommen, das Unternehmen stellt seinen Geschäftskunden Kundenkarten bereit, mit deren Hilfe verschiedene Abläufe für ihn vereinfacht werden können. Ein Service CardProcessing könnte gemeinsam mit dem oben beschriebenen Service CustomerManagement Bestandteil der Domäne Customer sein. Der eigentliche Gewinn dieser Strukturierung ist die Festlegung von Zuständigkeiten: Domänen-Verantwortliche entwerfen und pflegen das Service-Angebot für ihre Domäne.

Sie sind an einem ausgewogenen Angebot über die Realisierung eines einzelnen Umsetzungsprojekts hinweg interessiert und sind Partner des Service-Architekten. Die organisatorische Etablierung der Domänen-Verantwortlichen im Unternehmen ist von entscheidender Bedeutung. Sie müssen mit entsprechenden Befugnissen und Steuerungsmöglichkeiten ausgestattet werden.

Intra-Domänen-Aspekte beschreiben Fragen des internen Domänen-Aufbaus. Die *Service Provider*, die Dienste einer Domäne bereitstellen, können völlig unabhängig entwickelt worden sein, unterschiedliche Sichten auf Businessobjekte implementieren oder redundante Daten halten. Aufgabe der Intra-Domänen-Architektur ist es, mit einer Integrationsschicht ein domänen-internes Businessobjekt-Modell herzustellen, über dem alle Services der Domäne arbeiten können. Dies kann durch eine zusätzliche Persistenzschicht innerhalb der Domäne oder über die Einbindung von (XML)-Transformationen und Konvertierungen geschehen. Die Vorteile einer serviceorientierten Architektur könnten durch die Einführung von Intra-Domänen-Services auch innerhalb einer Domäne genutzt werden: Sie sind nur dort sichtbar und bieten Dienste zur Umsetzung des gemeinsamen Businessobjekt-Modells an.

Das plattform-spezifische Modell

Für die tatsächliche Umsetzung der entworfenen serviceorientierten Architektur muss eine geeignete Plattform ausgewählt werden. Folgende Fragen können als Entscheidungshilfe für die Auswahl einer Plattform herangezogen werden bzw. dabei helfen, Anforderungen an eine zu entwickelnde Plattform zu formulieren:

- Wie werden Services publiziert und vom *Service Requester* aufgefunden?
- Wie wird der Lebenszyklus eines Service unterstützt? Gibt es eine Service-Versionierung?
- Muss der *Service Requester* das Service-Interface exakt bedienen oder erhält er Unterstützung durch Transformationen?
- Welche Interaktionsstile unterstützt die Plattform?
- Welches Authentifizierungs- und Autorisierungskonzept wird angeboten?
- Welche Systemvoraussetzungen werden an anzubindende Agenten gestellt?
- Welche Kosten verursacht die Anbindung eines Agenten als *Service Requester* oder *Service Provider*?
- Wie verbreitet ist die Plattform?

Web-Services werden heute vielfach zur Kapselung von Altanwendungen eingesetzt. Für die Umsetzung unternehmensweiter serviceorientierter Architekturen mit Web-Services müssen die aktuellen Standardisierungsbestrebungen, vor allem auf dem Gebiet der *WS-Enhancements* (*WS-Security*, *WS-Policy*, *WS-Reliability* etc.), weiter fokussiert, vorangetrieben und durch Implementierungen unterstützt werden. Unternehmen können jedoch nicht immer darauf warten. Unternehmensspezifische SOA-Plattformen haben den Vorteil, spezielle Anforderungen, z.B. bezüglich Nutzer- und Rechteverwaltung, realisieren zu können. Ein langfristiger Investitionsschutz besteht aber nur dann, wenn von vornherein großer Wert auf die Verwendung anerkannter Standards und Erweiterbarkeit gelegt wird. Ein günstiger Migrationspfad zu Web-Services eröffnet sich, wenn die unternehmensspezifische SOA-Plattform auch das Auffinden und Aufrufen von Web-Services sowie den Einsatz von Web-Service-Clients als *Service Requester* erlaubt.

Fazit

Es stellt sich nun natürlich die Frage, wo denn in dem hier exerzierten Beispiel ein ►

Wert gestiftet wird. In vielen Organisationen existieren derzeit keine so disjunkten Bereiche für Reklamationsannahme (Callcenter) und -bearbeitung (Fachabteilung, Sachbearbeiter). Eine solche ist aber ökonomisch sinnvoll, da die teure Expertenzeit nicht für die zeitaufwändige Bearbeitung von Telefonaten, sondern zur Behebung des eigentlichen Kundenproblems verwendet werden kann. Eine SOA wie die hier gezeigte bildet eine optimale IT-Unterstützung für eine solche Aufgabenverteilung, die sogar ein Outsourcing des Callcenters ermöglicht.

Durch die Hinzunahme der Business-Intelligence-Komponente können in Echtzeit Statistiken über aktuelle Probleme und

offene Beschwerden erstellt und direkte Steuerungsmaßnahmen abgeleitet werden. Wichtig dabei ist, dass durch die mittels SOA gewonnene Entkopplung von Systemkomponenten die Erweiterung des Geschäftsprozesses um nebenläufige Protokollierung ohne jeglichen Eingriff in den Prozess selbst möglich geworden ist.

Nicht zuletzt verhilft die Einführung einer SOA zu einer klareren Strukturierung von Verantwortungsbereichen und ihrer Entsprechung in den IT-Systemen. Synergieeffekte und Konsolidierungspotenziale können somit nicht nur entdeckt, sondern auch mit Unterstützung der systemtechnischen Unternehmensarchitektur selbst ausgenutzt werden. ■

Literatur & Links

[Blo03] J. Bloomberg, Principles of SOA, in: Application Development Trends, Nr. 3, Vol. 10, S. 22

[Boo03] D. Booth et al., Web Services Architecture, W3C Working Draft 8 August 2003, siehe:

www.w3.org/TR/2003/WD-ws-arch-20030808/

[BPE03] Business Process Execution Language for Web Services Version 1.1 – BPEL4WS V1. 1, 5.5.03, siehe: www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[Chi03] R. Chinnici et al., Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language, W3C Working Draft, 11.6.03, siehe:

www.w3.org/TR/2003/WD-wsdl12-20030611

[Chr01] E. Christensen et al., Web Services Description Language (WSDL) 1.1, W3C Note 15.3.01, siehe:

www.w3.org/TR/2001/NOTE-wsdl-20010315

[Fow99] M. Fowler, UML Distilled. A Brief Guide to the Standard Object Modeling Language, Addison-Wesley Professional 1999