https://doi.org/10.20378/irbo-51758

14 Entwurf partieller SOA auf der Grundlage von Geschäftsprozessmodellen

Andree Krücke, Elmar J. Sinz

Zusammenfassung. In vielen Fällen soll nicht die komplette Anwendungssystem-Landschaft (AwS-Landschaft) eines Unternehmens gemäß dem Konzept der service-orientierten Architektur (SOA) gestaltet oder transformiert werden. Vielmehr ist es das Ziel, Teile davon, die hohen Flexibilitätsanforderungen ausgesetzt sind, auf Basis von SOA zu realisieren, während eher stabile Bereiche mit herkömmlichen AwS abgedeckt werden. Aus Sicht der AwS-Landschaft entsteht dadurch eine partielle SOA. Quelle der Flexibilitätsanforderungen an AwS sind korrespondierende Anforderungen an die von den AwS zu unterstützenden Geschäftsprozesse.
Der vorliegende Beitrag stellt einen modellbasierten Ansatz zum fachlichen Entwurf partieller SOA auf der Grundlage von Geschäftsprozessmodellen vor. Ausgehend von einem Geschäftsprozessmodell, in dem Bereiche mit besonders hohen Flexibilitätsanforderungen identifiziert wurden, führt der Ansatz in fünf Schritten zur Spezifikation eines SOA-AwS als Teil einer partiellen SOA. Die Schnittstellen zwischen dem SOA-AwS und den umgebenden AwS werden dabei explizit berücksichtigt. Der Ansatz wird anhand eines Fallbeispiels aus einem Logistikszenario (siehe E-Car-Szenario, Kapitel 2) demonstriert.

14.1 Problemstellung

Mit service-orientierten Architekturen (SOA) werden im Bereich betrieblicher Anwendungssysteme (AwS) insbesondere drei Ziele verfolgt: (1) Verbesserung der Wiederverwendbarkeit von Services und zugehörigen Software-Komponenten (Krafzig et al. 2005, S. 244f; Erl 2008, S. 72). (2) Erhöhung der Flexibilität von AwS (Melzer 2010, S. 33; Eckert et al. 2010, S. 18). Beide Ziele unterstützen wiederum (3) das IT/Business-Alignment sowohl auf strategischer als auch auf operativer Ebene (vgl. Aier und Winter 2009; Wagner und Weitzel 2006).

Grundlage von SOA sind Software-Komponenten, die bestimmte Services erbringen. Die Software-Komponenten sind lose gekoppelt, d. h. sie interagieren über den Austausch von Nachrichten. Das Zusammenwirken der Services im Rahmen von Prozessen wird in Form von hierarchischer Koordination (Orchest-

rierung) oder nicht-hierarchischer Koordination (Choreographie) beschrieben (Peltz 2003).

Viele Arbeiten zur Einführung von SOA in Unternehmen verfolgen einen globalen Ansatz, d. h. die Einführung von SOA oder die Transformation der bestehenden AwS-Architektur in Richtung SOA wird für das gesamte betriebliche System oder wenigstens ein größeres Teilsystem angestrebt. Hierfür werden sogenannte Enterprise Services identifiziert, aus denen dann die betrieblichen AwS konfiguriert werden (Schelp und Winter 2008, S. 9; Woods und Mattern 2006, S. 20). Bei diesen Arbeiten steht das Ziel der Wiederverwendbarkeit im Vordergrund, das Ziel der Flexibilität wird eher nachrangig verfolgt.

Der vorliegende Beitrag geht davon aus, dass in vielen Fällen nicht die komplette AwS-Landschaft eines Unternehmens gemäß SOA gestaltet bzw. transformiert werden soll. Vielmehr wird das Ziel verfolgt, diejenigen Teile der AwS-Landschaft, die besonders hohen Flexibilitätsanforderungen ausgesetzt sind, auf Basis von SOA zu realisieren (SOA-AwS) und andere, eher stabile Bereiche weiterhin mit herkömmlichen AwS zu unterstützen. Dabei stellt sich die Frage nach den Schnittstellen zwischen den SOA-AwS und den herkömmlichen AwS. Diese Schnittstellen betreffen einerseits den Übergang zwischen SOA-AwS und herkömmlichen AwS im Rahmen der Vorgänge und Vorgangsnetze zur betrieblichen Aufgabenerfüllung, sowie andererseits die Bereitstellung der von den SOA-AwS benötigten Daten, welche zum Teil von den herkömmlichen AwS verwaltet werden. Letzterer Aspekt wird aus methodischer Sicht in der aktuellen SOA-Literatur weitgehend vernachlässigt (Sinz 2008, S. 71–72; Carey 2008, S. 92).

Gegenstand des vorliegenden Beitrags ist die Entwicklung eines Ansatzes zur Unterstützung des Entwurfs partieller SOA für betriebliche AwS-Landschaften. Der Ansatz ist insbesondere durch folgende Merkmale charakterisiert:

- Explizite Ausrichtung auf partielle SOA, d. h. es steht das Ziel der Erhöhung der Flexibilität für ausgewählte Teile der AwS-Landschaft im Vordergrund. Über die Wiederverwendbarkeit der entwickelten Services und zugehörigen Software-Komponenten im Rahmen einer unternehmensweiten SOA werden keine Aussagen getroffen.
- Ausrichtung auf die fachliche Ebene von SOA. Die softwaretechnische Implementierung der Services, Komponenten und Schnittstellen wird nicht betrachtet.
- Die Entwicklung der SOA-AwS und die Spezifikation der fachlichen Schnittstellen zu den umgebenden AwS erfolgt modellbasiert auf der Basis

von Geschäftsprozessmodellen (GP-Modellen) gemäß dem Semantischen Objektmodell (SOM).

Der weitere Beitrag ist wie folgt aufgebaut: Abschnitt 14.2 behandelt Ansätze aus der Literatur zur Entwicklung von SOA. Abschnitt 14.3 stellt den Ansatz zum fachlichen Entwurf von SOA-AwS als Bestandteil einer partieller SOA vor. Die Anwendung des Ansatzes wird in Abschnitt 14.4 anhand eines Fallbeispiels aus einem e-Car-Szenario demonstriert. Eine Diskussion des Ansatzes in Abschnitt 14.5 schließt den Beitrag ab.

14.2 Ansätze zur Entwicklung service-orientierter Architekturen

Vorrangige Ziele, die mit SOA verfolgt werden, sind eine Verbesserung der Wiederverwendbarkeit und eine Erhöhung der Flexibilität auf IT-Ebene. Wesentliche Voraussetzung für eine breite Wiederverwendbarkeit von Services und zugehörigen Software-Komponenten ist eine unternehmensweit ausgerichtete SOA-Strategie (z. B. Kavianpour 2007, S. 531f). Ansätze zur Entwicklung von SOA stützen sich hierzu i. d. R. auf eine Vorgehensstrategie, bei der die unternehmensweite (Weiter-)Entwicklung und Ausbreitung einer SOA schrittweise erfolgt (Krafzig et al. 2005, S. 11; Masak 2005, S. 221ff). Dieses inkrementelle Vorgehen hilft, die aus einer unternehmensweiten SOA-Strategie abgeleitete Entwicklungsaufgabe sinnvoll zu untergliedern, kann aber nicht verhindern, dass Systeme mit zunehmender Größe aufgrund der Gesamtkomplexität schwer beherrschbar und überschaubar werden (Masak 2007, S. 323ff). Die Handhabung dieser Komplexität ist nicht trivial und kann zur Ernüchterung in Bezug auf den erhofften Nutzen von SOA führen.

Neben der höheren Komplexität einer auf Wiederverwendung ausgerichteten AwS-Landschaft wirken noch weitere Aufwandstreiber negativ auf den wahrgenommenen Nutzen von SOA. So sind z. B. Rahmenbedingungen in der IT-Organisation einer Unternehmung zu schaffen, damit wiederverwendbare Software-Komponenten überhaupt erst verwaltet, gefunden und eingesetzt werden können (Krafzig et al. 2005, S. 7; Masak 2007, S. 135ff).

Der Nutzen einer umfassenden Wiederverwendbarkeit von Software-Artefakten hingegen besteht zum einen im mehrfachen Einsatz von einmal entwickelten Programmfunktionen, zum anderen aber auch in der Reduzierung von Redundanzen und Inkonsistenzen auf Ebene der Unternehmensdaten (Krafzig et al. 2005, S. 244f).

Die Ablösung eines Altsystems durch eine auf Wiederverwendbarkeit abzielende Service-Architektur führt im Allgemeinen gleichzeitig zu einer Erhöhung der Flexibilität des Systems. Zu hinterfragen ist hierbei jedoch, ob die gewonnene Flexibilität auch in jedem Fall einen Nutzen darstellt. Sofern ein Altsystem zur Unterstützung eines über die Zeit hinweg relativ stabilen Aufgabenbereichs dient, dürften der Flexibilitätsbedarf und damit auch der durch die Systemumstellung erreichbare Nutzen eher gering, der damit verbundene Aufwand aber eher hoch sein. Bei Berücksichtigung aller genannten Faktoren ist somit davon auszugehen, dass Wiederverwendbarkeit zwar i. d. R. ein wünschenswertes Ziel darstellt, unter Berücksichtigung des damit verbundenen Aufwands aber nicht immer erstrebenswert ist, bzw. nicht im gleichen Maße priorisiert wird, wie dies beim Ziel der Flexibilitätserhöhung der Fall ist (siehe auch Berlecon Research 2006, S. 4). So kann der Flexibilitätsbedarfs eines Unternehmens, der sich z. B. durch raschen Wandel von Geschäftsprozessen ergibt, aus Sicht des Unternehmens hoch sein und somit den Einsatz einer auf das Ziel der Erhöhung von Flexibilität ausgerichteten SOA begründen, ohne dass das Ziel der umfassenden Verbesserung von Wiederverwendbarkeit verfolgt werden muss. Hierbei ist davon auszugehen, dass durch Betonung des Flexibilitätsaspekts gegenüber der Wiederverwendbarkeit auch die Reichweite einer SOA von einer unternehmensweiten Sicht hin zu einer eher geschäftsprozessorientierten Sicht wechselt, da diese den Bezugspunkt für Flexibilitätsanforderungen darstellt.

Eine wesentliche Aufgabe zur Erreichung des Flexibilitätsziels ist die Ableitung von Services ausgehend von den zu (teil-)automatisierenden Geschäftsprozessen (Schlimm 2010, S. 286). Diese Aufgabe steht im Mittelpunkt vieler Ansätze zur Entwicklung von SOA (z. B. Kätker und Patig 2009; Bernhard und Jahn 2009), wobei oftmals eine Unterscheidung zwischen fachlicher und technischer Serviceidentifizierung stattfindet (Thomas et al. 2009). Während die fachliche Sicht mit einer Ableitung von Services aus Geschäftsprozesssicht korrespondiert, unterstützt die technische Sicht die Identifizierung von Services unter Berücksichtigung der vorhandenen AwS-Landschaft eines Unternehmens. Hinsichtlich des angestrebten Flexibilitätsziels sind beide genannten Sichten relevant, was bei einer Analyse des Flexibilitätsbedarfs von Teilbereichen eines Geschäftsprozesses deutlich wird. So gibt es i. d. R. Bereiche eines Geschäftsprozesses mit hohem Flexibilitätsbedarf, während andere Bereiche als eher stabil klassifizierbar sind und somit aus Aufwandssicht durch schon vorhandene AwS im Unternehmen unterstützt werden können, die ggfs. eine eher unflexible IT-Architektur aufweisen.

Durch eine Prozessschnittstelle können diese AwS mit einem SOA-basierten AwS gekoppelt werden, das der Unterstützung von Bereichen des Geschäftsprozesses mit hohem Flexibilitätsbedarf dient. Bestehende Ansätze zur Entwicklung von SOA unterstützen die Einbindung bestehender AwS in eine SOA i. d. R. dadurch, dass das einzubindende AwS aus Sicht der SOA als Servicedienstleister gesehen wird (z. B. Alahmari et al. 2010), der über eine Prozessschnittstelle seine Dienste in die SOA einbringen kann. Ein Entwurfsmuster zur Ableitung einer solchen Prozessschnittstelle für Legacy Systeme ist z. B. das *Legacy Wrapper Design Pattern* nach Erl (Erl 2009, S. 441ff). Ausgeblendet wird hierbei jedoch, dass das einzubindende System auch aus Sicht der gesamten AwS-Landschaft betrachtet werden muss. Aus dieser Sicht stellt es keinen Servicedienstleister im Rahmen der SOA dar, sondern ein herkömmliches AwS, das im operativen Betrieb eines Unternehmens genutzt wird und ggfs. auf den selben Daten operiert, die auch im Rahmen der SOA referenziert werden. Wird eine SOA, in die vorhandene AwS zur Service-Erbringung eingebunden werden, nicht als Hybrid-System (Hutchinson et al. 2008, S. 35ff) angesehen, d. h. als Verknüpfung von AwS mit ggfs. überlappenden Datenstrukturen, so ist die Gefahr inkonsistenter Daten und damit auch inkonsistenter Prozesse nicht auszuschließen.

Die obigen Ausführungen machen deutlich, dass eine auf Geschäftsprozesse mit partiell hohem Flexibilitätsbedarf ausgerichtete SOA-Strategie spezifische Anforderungen an die eingesetzten Ansätze zur SOA-Entwicklung stellt. Diese Anforderungen werden in Teilen von bestehenden Ansätzen in der Literatur zwar berücksichtigt (zum Vergleich serviceorientierter Vorgehensmodelle siehe z. B. auch Thomas et al. 2009), eine ganzheitlich auf die Bedürfnisse der genannten SOA-Strategie ausgerichtete Vorgehensweise findet sich jedoch nicht. Der im Folgenden entwickelte Ansatz leistet einen Beitrag zur Schließung dieser Lücke, indem er ausgehend von einem GP-Modell und unter besonderer Berücksichtigung von Flexibilitätsanforderungen innerhalb einzelner Geschäftsprozesse die Ableitung einer partiellen fachlichen SOA modellbasiert unterstützt - unter besonderer Berücksichtigung der zugrunde liegenden Datenstrukturen.

14.3 Fachlicher Entwurf partieller SOA auf der Grundlage von SOM-GP-Modellen

Im Folgenden wird der im Mittelpunkt des Beitrags stehende Ansatz zum fachlichen Entwurf von SOA-AwS als Teil einer partieller SOA auf der Grundlage von

SOM-Geschäftsprozessmodellen vorgestellt. Ausgehend von einer kurzen Einführung in das Semantische Objektmodell (Abschnitt 14.3.1) wird das verwendete SOA-Architekturmodell beschrieben (14.3.2). Anschließend wird die methodische Konzeption des Ansatzes erläutert (14.3.3).

14.3.1 Semantisches Objektmodell (SOM)

Das Semantische Objektmodell (SOM) ist eine objekt- und geschäftsprozessorientierte Methodik zur ganzheitlichen Modellierung betrieblicher Systeme (Ferstl und Sinz 2008, S. 192ff.). Den Kern der SOM-Methodik bildet ihre Unternehmensarchitektur, welche drei Modellebenen umfasst: (1) Der Unternehmensplan beschreibt die Gesamtaufgabe des Unternehmens aus Außenperspektive und spezifiziert zugehörige Strategien und Rahmenbedingungen. (2) Das Geschäftsprozessmodell beschreibt das Lösungsverfahren für die Realisierung des Unternehmensplans aus einer Innenperspektive auf Struktur und Verhalten der Teilaufgaben des Unternehmens. (3) Das Ressourcenmodell beschreibt ebenfalls aus Innenperspektive die personellen und maschinellen Aufgabenträger zur Durchführung der betrieblichen Aufgaben. Die maschinellen Aufgabenträger werden durch die AwS-Landschaft des Unternehmens gebildet.

Im Rahmen des vorliegenden Beitrags werden das Geschäftsprozessmodell der zweiten Modellebene sowie die fachlichen AwS-Spezifikationen der dritten Modellebene genutzt. Im Vergleich zu anderen Ansätzen der Geschäftsprozessmodellierung, wie z. B. zu den eher ablauforientierten Ansätzen *Ereignisgesteuerte Prozesskette* (*EPK*) (Keller et al. 1992) und *Business Process Model and Notation* (*BPMN*) (Object Management Group (OMG) 2009) besitzt SOM die Eigenschaft, dass Geschäftsprozessmodelle sowohl aus Struktursicht als auch aus Verhaltenssicht modelliert werden und aus diesen beiden Sichten initiale fachliche Spezifikationen der zugehörigen AwS modellbasiert abgeleitet werden können.

Die Struktursicht eines SOM-Geschäftsprozessmodells (Ferstl und Sinz 1995) wird in Form eines Interaktionsschemas (IAS) modelliert. Dieses enthält betriebliche Objekte, die durch betriebliche Transaktionen koordiniert werden. Ein betriebliches Objekt umfasst eine Menge von Aufgaben, die auf einem gemeinsamen Aufgabenobjekt durchgeführt werden, mit zugehörigen Sach- und Formalzielen. Es werden zwei Koordinationsformen unterschieden, eine hierarchische Koordination unter der Nutzung von Steuer- und Kontrolltransaktionen (S, K) und eine nicht-hierarchische Koordination mithilfe von Anbahnungs-, Vereinbarungs- und Durchführungstransaktionen (A, V, D) (Ferstl und Sinz 2008, S. 66ff).

Die zugehörige Verhaltenssicht eines SOM-Geschäftsprozessmodells wird in einem Vorgangs-Ereignis-Schema (VES) dargestellt. Dieses beschreibt den ereignisgesteuerten Ablauf der den betrieblichen Objekten zugeordneten Aufgaben. Ereignisbeziehungen zwischen Aufgaben unterschiedlicher Objekte werden in Form von Transaktionen modelliert, Ereignisbeziehungen zwischen Aufgaben desselben Objekts in Form von objektinternen Ereignissen.

Aus dem IAS und dem VES kann die initiale fachliche Spezifikation der AwS zur Automatisierung der modellierten Geschäftsprozesse abgeleitet werden. Diese Spezifikation umfasst ein Konzeptuelles Objektschema (KOS) und ein zugehöriges Vorgangsobjektschema (VOS). Das KOS spezifiziert das Aufgabenobjekt der in den betrieblichen Objekten zusammengefassten Aufgaben in Form einer Menge von in Beziehung stehenden Konzeptuellen Objekttypen (KOT). Die KOT korrespondieren mit den betrieblichen Objekten und Transaktionen des Geschäftsprozessmodells. Das VOS spezifiziert eine Menge von in Beziehung stehenden Vorgangsobjekttypen (VOT). Jeder VOT korrespondiert mit genau einer betrieblichen Aufgabe und beschreibt das Zusammenwirken von KOTs bei der Durchführung dieser Aufgabe.

14.3.2 SOA-Architekturmodell

Die Software-Architektur eines AwS umfasst dessen Bauplan im Sinne einer Spezifikation von Komponenten und Beziehungen unter allen relevanten Blickwinkeln sowie die Konstruktionsregeln für die Erstellung des Bauplans (Sinz 2002, S. 1055). Die allgemeine Spezifikation von Komponenten- und Beziehungstypen wird zusammen mit zugehörigen Konstruktionsregeln als Software-Architekturmodell bezeichnet. Dieses stellt Strukturvorgaben für die Erstellung der konkreten Software-Architekturen einer Klasse von AwS bereit.

Software-Architekturmodelle sind auf bestimmte Entwurfsziele ausgerichtet. Mit SOA-Architekturmodellen werden im Allgemeinen insbesondere die Ziele Wiederverwendbarkeit und Flexibilität verfolgt. Wie bereits genannt, steht in diesem Beitrag das Ziel der Flexibilität im Vordergrund. Eine wichtige Konstruktionsregel für flexible AwS betrifft die nachrichtenbasierte, lose Kopplung der Komponenten (Papazoglou 2008, S. 16f). Das Prinzip der losen Kopplung wird auch bei der Verknüpfung betrieblicher Objekte in SOM-GP-Modellen angewandt (Ferstl und Sinz 2008, S. 199ff). Auf dieses Merkmal nimmt das im Folgenden beschriebene und in Abb. C-71 dargestellte SOA-Architekturmodell Bezug.

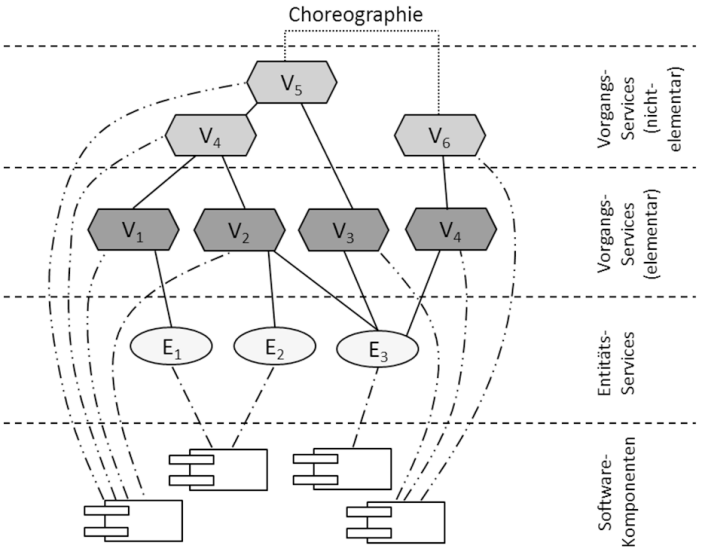


Abb. C-71: SOA-Architekturmodell

Elementare Bausteine einer SOA-Architektur sind Services und Komponenten. Ein Service stellt eine definierte Leistung dar, die als Element eines oder mehrerer größerer Verarbeitungsabläufe verwendet werden kann. Zudem liegt einem Service eine vertragliche Übereinkunft zwischen Serviceanbieter und -nutzer zugrunde (Richter et al. 2005, S. 413). Komponenten dienen der Erbringung eines oder mehrerer Services, wobei sie die Implementierung der Services gegenüber der die Services nutzenden Umgebung verbergen. Im allgemeinen Sprachgebrauch werden die Begriffe Service und Komponente oftmals nicht scharf unterschieden (Siedersleben 2007, S. 112), was vor allem aus folgendem Grund kritisch zu bewerten ist: Aufgabenobjekte im Sinne von Datenstrukturen, die zur Erbringung eines Services erforderlich sind, werden nicht von einem Service, sondern von der zugrunde liegenden Komponente verwaltet. Liegt nun zwei unterschiedlichen Services zum Teil ein gemeinsames Aufgabenobjekt zugrunde, so ist der konsistente Zustand der Aufgabenobjekte sicherzustellen. Sofern die Services von zwei unterschiedlichen Komponenten erbracht werden, ist zur Wahrung der Konsistenz ggfs. eine Datenschnittstelle auf Komponentenebene zu berücksichtigen - ein Erfordernis, das sich aus reiner Servicebetrachtung nicht ableiten ließe.

In der Literatur wurde eine Reihe von Vorschlägen für SOA-Architekturmodelle unterbreitet (Leyking et al. 2007, S. 183). Ein Beispiel ist der Vorschlag von Hess et al., in dem Bestands-, Prozess-, Funktions- und Interaktionskomponenten unterschieden werden (Hess et al. 2006, S. 398). Letztere stellen dabei Services für die Interaktion mit Nutzern als personelle Aufgabenträger dar. Im vorliegenden Beitrag wird aus Gründen der Vereinfachung die Frage der Teilautomatisierung von Geschäftsprozessen und damit die Interaktion mit personellen Aufgabenträgern nicht betrachtet; dies bleibt einer Weiterentwicklung des Ansatzes vorbehalten. Bezüglich der weiteren Komponenten erfolgt die Differenzierung anhand der von ihnen erbrachten Servicekategorien. Die Servicekategorien weisen einen Bezug zur fachlichen Granularität auf (Heutschi 2007, S. 172) und bestimmen Konstruktionsregeln für die service-erbringenden Komponenten.

Es werden zwei Kategorien von Services unterschieden:

1. Entitäts-Services: Grundlage für die Automatisierung von Geschäftsprozessen durch AwS ist die rechnergestützte Verwaltung der den betrieblichen Aufgaben zugrunde liegenden Aufgabenobjekte. Gemäß SOM-Methodik wird das Aufgabenobjekt einer Aufgabe in Form von in Beziehung stehenden KOTs spezifiziert. Die Schnittstelle zu den KOTs wird in Form eines Entitäts-Service realisiert. Die dem Service zugrundeliegende Komponente kapselt und persistiert ggfs. die Attribute der zu verwaltenden KOTs, der Service selbst ermöglicht den externen Zugriff auf die Attribute und Funktionen des Aufgabenobjekts. Die Operatoren des Services korrespondieren mit einer fachlichen Sicht auf das Aufgabenobjekt, so dass z. B. aus Geschäftssicht zusammenhängende Attribute des Aufgabenobjekts auch als Einheit durch einen einzigen Serviceaufruf abgefragt werden können (Josuttis 2007, S. 63).
2. Vorgangs-Services: Die Steuerung des Zusammenwirkens von KOTs bei der automatisierten Durchführung einer betrieblichen Aufgabe entspricht einem Vorgang. Zur Automatisierung der Koordination von Vorgängen wird die Service-Kategorie der Vorgangs-Services bereitgestellt. Ein elementarer Vorgangs-Service orchestriert im Rahmen der Durchführung einer betrieblichen Aufgabe ggfs. mehrere Entitäts-Services. Ein nicht-elementarer Vorgangs-Services orchestriert das mit den Aufgaben eines betrieblichen Objekts korrespondierende Vorgangsnetz, welches elementare und ggfs. weitere nicht-elementare Vorgangs-Services umfasst, oder nimmt an Choreografien mit benachbarten Vorgangs-Services teil.

Anhand der beiden Servicekategorien erfolgt die Bildung der die Services erbringenden Komponenten. Die Konstruktionsregel lautet, dass fachlich zusammenhängende Aufgabenobjekte, die durch eine Menge von in Beziehung stehenden KOTs repräsentiert werden und auf die über entsprechende Entitäts-Services zugegriffen werden kann, zu einer Komponente zusammengefasst werden. Die Erbringung fachlich in Beziehung stehender elementarer und/oder nicht-elementarer Vorgangs-Services wird ebenfalls in einer Komponente zusammengefasst. Als Kriterium für die Abgrenzung der beiden Arten von Komponenten dient das Aufgabenobjekt bzw. das Vorgangsnetz eines betrieblichen Objekts.

Aus Orchestrierungssicht weisen nicht-elementare Vorgangs-Services die gröbste Granularität unter den vorgestellten Service-Kategorien auf. Zur Automatisierung des gesamten Geschäftsprozesses bedarf es aber auch einer Koordination zwischen den Vorgangsnetzen der beteiligten betrieblichen Objekte. Da betriebliche Objekte grundsätzlich autonom sind, erfolgt die Koordination der Vorgangs-Services unterschiedlicher betrieblicher Objekte gemäß dem Prinzip der Choreographie (Peltz 2003).

14.3.3 Konzeption des Ansatzes

Abb. C-72 zeigt das schrittweise Vorgehen zur modellbasierten Entwicklung einer partiellen SOA.

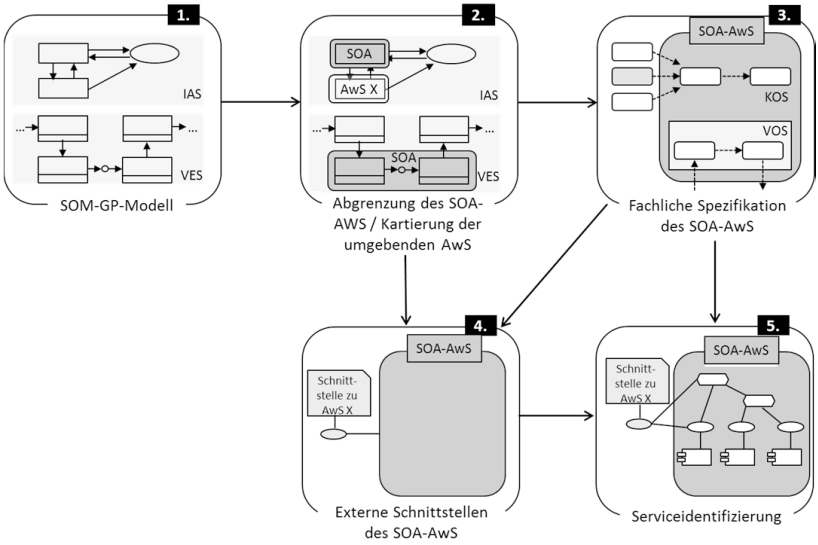


Abb. C-72: Modellbasierte Entwicklung einer partiellen SOA

Ausgangspunkt ist ein geeignet abgegrenztes SOM-GP-Modell. Ergebnis ist die Spezifikation der SOA für ein spezifisches Teil-AwS aus fachlicher Sicht (SOA-AwS) zusammen mit den externen Schnittstellen zu umgebenden AwS. Das Vorgehen wird zunächst allgemein erläutert. In Abschnitt 14.4 erfolgt die Anwendung des Ansatzes auf ein nicht-triviales Fallbeispiel.

Schritt 1: SOM-GP-Modell. Ausgangspunkt ist ein SOM-GP-Modell in Form eines IAS und eines korrespondierenden VES. Es wird angenommen, dass die modellierten Geschäftsprozesse weitgehend automatisiert durchgeführt werden. Dazu werden im Allgemeinen mehrere interagierende AwS eingesetzt. Weiter wird angenommen, dass in dem GP-Modell bereits ein Bereich mit hohen Flexibilitätsanforderungen identifiziert wurde, welcher durch ein SOA-AwS automatisiert werden soll, während umgebende, längerfristig eher stabile Bereiche durch herkömmliche, ggfs. schon existierende AwS unterstützt werden. Das SOM-GP-Modell ist so abgegrenzt, dass es die durch das SOA-AwS zu automatisierenden Aufgaben sowie umgebende Aufgaben umfasst.

Schritt 2: Abgrenzung des SOA-AwS / Kartierung der umgebenden AwS. Gemäß der SOM-Methodik werden die Aufgaben eines betrieblichen Objekts durch genau ein AwS automatisiert. Umgekehrt kann ein AwS mehrere betriebliche Objekte unterstützen. Durch diese Forderung wird sichergestellt, dass sich die fachlichen Beziehungen zwischen AwS vollständig aus dem GP-Modell her-

aus erklären lassen. Ggfs. ist ein betriebliches Objekt zur Erfüllung der Forderung weiter zu verfeinern. In Schritt 2 werden diejenigen betrieblichen Objekte im GP-Modell abgegrenzt, deren Aufgaben aufgrund der Flexibilitätsanforderungen durch das SOA-AwS automatisiert werden sollen. Den umgebenden betrieblichen Objekten, deren Aufgaben durch herkömmliche AwS automatisiert werden sollen, werden die jeweiligen AwS zugeordnet. Die Beziehungen zwischen dem SOA-AwS und den umgebenden AwS lassen sich aus dem GP-Modell anhand der Transaktionen zwischen den jeweiligen betrieblichen Objekten ableiten. Durch die Zuordnung zu betrieblichen Objekten werden die einzelnen AwS im GP-Modell kartiert.

Schritt 3: Fachliche Spezifikation des SOA-AwS. Für den Ausschnitt des GP-Modells, dessen betriebliche Objekte durch das SOA-AwS unterstützt werden sollen, wird nun die fachliche Spezifikation des SOA-AwS in Form eines VOS und eines KOS aus dem GP-Modell abgeleitet (Ferstl und Sinz 2008, S. 215ff). Die VOTs des VOS und die KOTs des KOS bilden anschließend die Grundlage für die modellbasierte Identifikation der Services des SOA-AwS sowie der externen Schnittstellen zu den umgebenden AwS.

Schritt 4: Externe Schnittstellen des SOA-AwS. Auf der Grundlage der Ergebnisse aus Schritt 2 und Schritt 3 werden die externen Schnittstellen des SOA-AwS aus fachlicher Sicht modellbasiert abgeleitet. Die Kartierung aus Schritt 2 zeigt anhand der Transaktionen zwischen betrieblichen Objekten auf, welche Prozessschnittstellen zwischen dem SOA-AwS und den umgebenden AwS vorzusehen sind. Diese externen Schnittstellen werden entsprechend der vorgestellten SOA-Architektur durch Vorgangs-Services realisiert, die von den umgebenden AwS bereitzustellen sind. Zur Spezifikation dieser Schnittstellen wird das in Schritt 3 spezifizierte VOS herangezogen: Alle Nachrichtenflüsse, die von VOTs des SOA-AwS empfangen oder ausgelöst und bei denen der Nachrichtensender bzw. -empfänger ein externes AwS ist, sind durch eine externe Prozessschnittstelle zu unterstützen.

Neben der Berücksichtigung externer Prozessschnittstellen ist auch das Erfordernis externer Datenschnittstellen zwischen AwS zu prüfen. Hierzu sind die Aufgabenobjekte der durch eine betriebliche Transaktion verbundenen Aufgaben zu betrachten. Eine Datenschnittstelle ist dann erforderlich, wenn es Überlappungen zwischen den genannten Aufgabenobjekten gibt, wobei die betrieblichen Objekte, denen diese Aufgaben zugeordnet sind, von unterschiedlichen AwS unterstützt werden und zudem die Datenbestände, welche das gemeinsame Aufgabenobjekt repräsentieren, nur von einem der beiden AwS verwaltet werden sol-

len (Datenhoheit). Die Datenschnittstellen werden in Form von Entitäts-Services spezifiziert, die aus dem in Schritt 3 entwickelten KOS abgeleitet werden. Für die Spezifikation der Prozess- und Datenschnittstellen ist das Konzept des *Message Exchange Pattern* (*MEP*) hilfreich, welches die Kommunikationsart zwischen Servicenachfrager und Serviceanbieter beschreibt (W3C 2007). Relevante Ausprägungen des MEP sind das *In-Out MEP* und das *In-Only MEP*. Ersteres beschreibt eine bidirektionale Kommunikation zwischen Serviceaufrufer und –anbieter (Request-Response), letzteres eine unidirektionale Kommunikation, bei welcher der Serviceaufrufer keinen Rückgabewert durch den Serviceanbieter erwartet. Die Kommunikationsart ist auch für die fachliche Spezifikation der Schnittstellen relevant. So werden z. B. unidirektionale Prozessschnittstellen, welche der seriellen Verknüpfung von Vorgangs-Services dienen, im Allgemeinen auf Basis des *In-Only MEP* gestaltet. Bidirektionale Prozessschnittstellen, über die ein Service aufgerufen und dessen Antwort unmittelbar erwartet wird, werden auf Basis *des In-Out-MEP* realisiert.

Schritt 5: Serviceidentifizierung. Im letzten Schritt wird auf Basis der Ergebnisse aus den Schritten 3 und 4 die Innensicht des SOA-AwS in Form einer fachlichen Service-Architektur festgelegt. Grundlage hierfür ist das in Abschnitt 14.3.2 vorgestellte SOA-Architekturmodell. Entitäts-Services werden anhand des Aufgabenobjekts betrieblicher Aufgaben identifiziert, welches durch in Beziehung stehende KOTs repräsentiert wird. Elementare Vorgangs-Services werden anhand der VOTs einzelner betrieblicher Aufgaben, nicht-elementare Vorgangs-Services anhand der Vorgangsnetze betrieblicher Objekte identifiziert.

Entsprechend der Ausrichtung des hier vorgestellten Ansatzes erfolgt die Spezifikation von SOA-AwS auf fachlicher Ebene. Implementierungsdetails, etwa in Form des Lösungsverfahrens von Services, obliegen hingegen dem Verantwortungsbereich des Service-Erbringers. Die Anwendung des „Separation-of-Concerns“-Prinzips (vgl. Erl 2009, S. 300ff.) bedeutet allerdings nicht, dass die fachliche SOA-AwS-Spezifikation auf Ebene der Realisierung einzelner Services lediglich als Vorgabe für zu erbringende Schnittstellen dient. Vielmehr kann sie auch bei der modellgetriebenen Ableitung der Vorgangssteuerung nicht-elementarer Services unterstützen. Anknüpfpunkt ist hierbei der in Kapitel 13 beschriebene Ansatz nach Pütz und Sinz, der die modellgetriebene Ableitung von objektspezifischen *BPMN*-Workflows aus SOM-GP-Modellen ermöglicht.

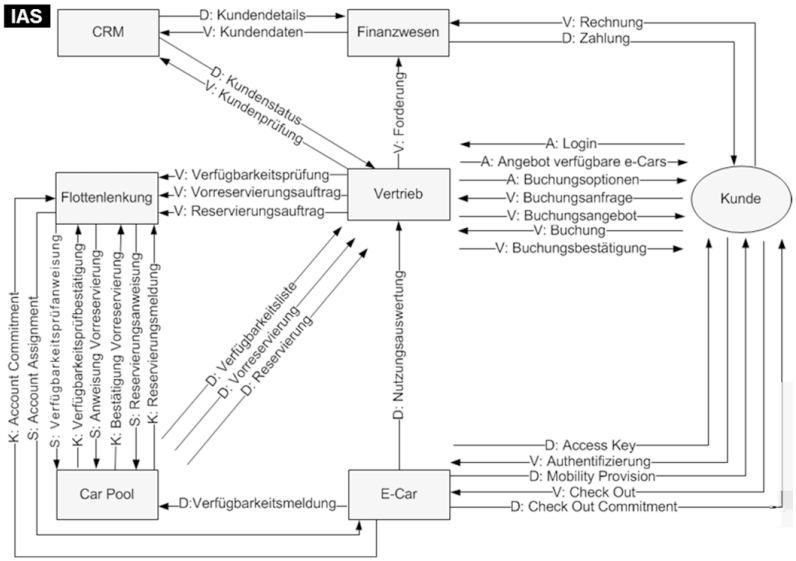
Naturgemäß bestehen bei der Abgrenzung von Services und bei der Zusammenfassung zu Komponenten Freiheitsgrade. Hier stellen aus inhaltlicher Sicht die fachlichen Strukturen des zu unterstützenden GP-Modells ein Entscheidungs-

kriterium bereit: Eine hohe Flexibilität der AwS setzt eine hohe Strukturähnlichkeit zwischen GP-Modell und AwS-Architektur voraus. Aus formaler Sicht ist die Granularität der Services und Komponenten zu berücksichtigen. Zu grob geschnittene Services behindern die Flexibilität des SOA-AwS, zu fein geschnittene erhöhen die strukturelle Komplexität des AwS und können u. a. negative Auswirkungen in Bezug auf die Performanz der späteren Implementierung des AwS besitzen (Bell 2010, S. 34).

14.4 Fallbeispiel

Im Folgenden wird die Anwendung des in Abschnitt 14.3.3 erläuterten Ansatzes anhand des Teilbereichs *Mobility Provision* des in Kapitel 2 vorgestellten fiktiven Wertschöpfungsnetzes *e-Car Net* demonstriert. Die Gesamtaufgabe des Teilbereichs *e-Car Net Mobility* ist die onlinegestützte Vermietung von Elektroautos (e-Cars) in ausgewählten deutschen Städten.

Schritt 1: SOM-GP-Modell. Ausgangspunkt ist das in Abb. C-73 und Abb. C-74 dargestellte Geschäftsprozessmodell der *e-Car-Net Mobility*, das durch eine AwS-Landschaft mit partieller SOA unterstützt werden soll. IAS und VES liegen auf detaillierten Zerlegungsstufen vor.



Objektzerlegung

E-Car-Net Mobility
- **Vertrieb**
- **Flotte**
  - **Flottenlenkung**
  - **Car Pool**
    - S: Verfügbarkeitsprüfanweisung
    - K: Verfügbarkeitsprüfbestätigung
    - D: Verfügbarkeitsliste
    - S: Anweisung Vorreservierung
    - K: Bestätigung Vorreservierung
    - D: Vorreservierung
    - S: Reservierungsanweisung
    - K: Reservierungsmeldung
    - D: Reservierung
  - **E-Car**
    - S: Account Assignment
    - K: Account Commitment
    - D.seq1: Nutzungsauswertung
    - D.seq2: Verfügbarkeitsmeldung
- **CRM**
  - D: Prüfung_Kundenstatus
    - D.V: Kundenprüfung
    - D.D: Kundenstatus
  - D: Bereitstellung_Kundendaten
    - D.V: Anforderung_Kundendaten
    - D.D: Kundendetails
- **Finanzwesen**

Kunde

Transaktionszerlegung

E-Car Rental
- **A: Angebot**
  - A.seq1: Login
  - A.seq2: kundenspezifisches Angebot
    - A.seq2.par1: verfügbare e-Cars
    - A.seq2.par2: Buchungsoptionen
- **V: Anmietung**
  - V.seq1: Buchungsanfrage
  - V.seq2: Buchungsangebot
  - V.seq3: Buchung
  - V.seq4: Buchungsbestätigung
- **D: Bereitstellung e-Car**
  - D.seq1: AccessKey
    - D.seq1.V: Authentifizierung
    - D.seq1.D: Mobility Providing
  - D.seq2: Mietrückgabe
    - D.seq2.V: Check_Out
    - D.seq2.D: Check_Out_Commitmen
  - D.seq3: Abrechnung
    - D.seq3.V: Rechnung
    - D.seq3.D: Zahlung

Abb. C-73: Interaktionsschema sowie Objekt- und Transaktionszerlegung für den Geschäftsprozess e-Car Rental

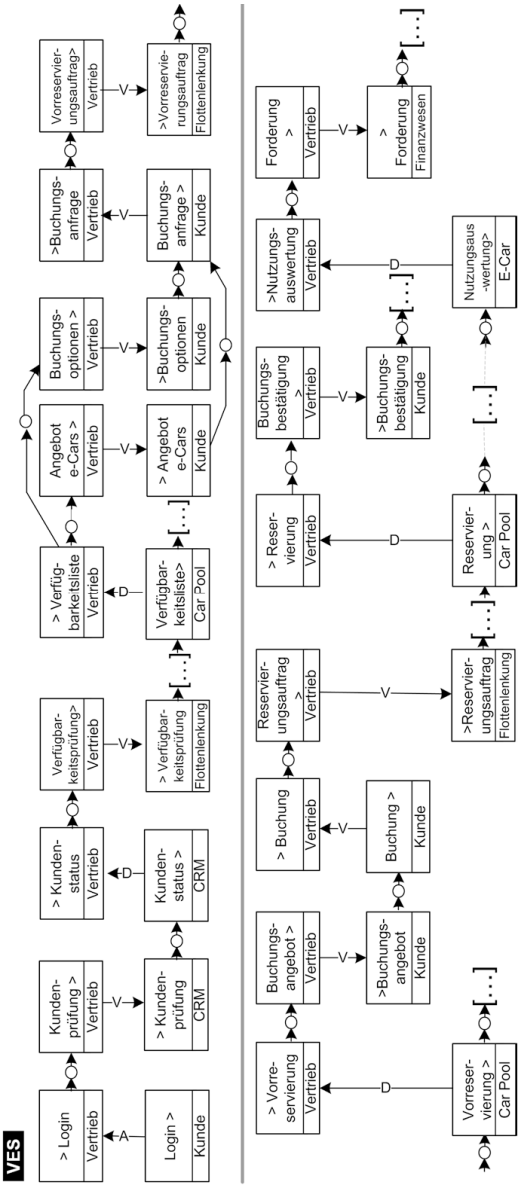


Abb. C-74: Vorgangsereignisschema für den Geschäftsprozess e-Car Rental

Ausgehend von der Transaktion *e-Car Rental*, welche die Leistung des Unternehmens *e-Car-Net Mobility* an Kunde übergibt, zeigt das Protokoll der Objekt- und Transaktionszerlegung in Abb. C-73 die einzelnen Zerlegungsschritte.

Innerhalb der Diskurswelt stellt *Vertrieb* das zentrale betriebliche Objekt und gleichzeitig die wichtigste Schnittstelle zum Kunden dar. Aufgabe von *Vertrieb* ist die Koordination der Leistungserbringung an den Kunden, wobei die einzelnen Teilleistungen selbst von weiteren autonomen Objekten erbracht werden. Dem betrieblichen Objekt *Customer Relationship Management (CRM)* obliegt hierbei die Verwaltung der Kundenstammdaten. *Finanzwesen* hat im Rahmen des Geschäftsprozesses die Aufgabe, den prozessbegleitenden Zahlungsverkehr mit dem Kunden abzuwickeln und zu überwachen. Die betrieblichen Objekte *Flottenlenkung*, *Car Pool* und *e-Car* dienen schließlich dem eigentlichen Flottenmanagement, d. h. der Verwaltung und Lenkung des Fuhrparks. Die Leistung e-Car Rental soll vollautomatisiert erbracht werden.

Schritt 2: Abgrenzung des SOA-AwS / Kartierung der umgebenden AwS. Gegeben sei eine Analyse der Flexibilitätsanforderungen an den Geschäftsprozess sowie eine Analyse der bestehenden AwS-Landschaft des Unternehmens, welche eine Zuordnung von AwS zu betrieblichen Objekten des GP-Modells ermöglicht (Abb. C-75).

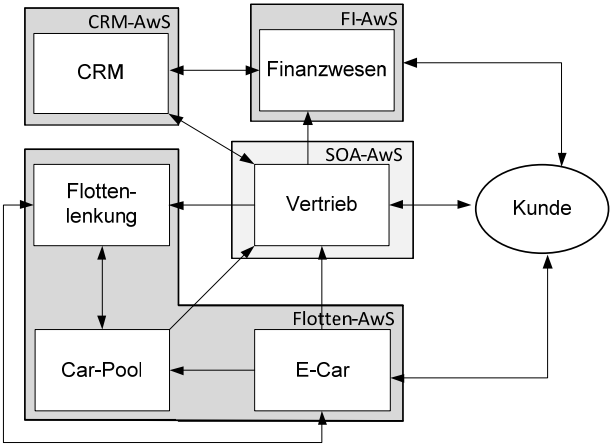


Abb. C-75: Abgrenzung und Kartierung des SOA-AwS

Vertrieb: Für das Objekt *Vertrieb* wurden hohe Flexibilitätsanforderungen identifiziert. Es ist geplant, dass künftig neben der hier betrachteten Leistung der Online-Direkt-Vermietung auch eine Vermietung über Call-Center, Zwischenmakler

oder per ad-hoc-Nutzung von e-Cars erfolgen soll. Ggfs. soll dabei auch auf andere externe Car Pools zugegriffen werden, weshalb die spätere Erweiterbarkeit des GP-Modells um zusätzliche Ausführungsvarianten in diesem Bereich zu berücksichtigen ist. Aus diesem Grund wird für den Vertrieb eine Unterstützung durch ein SOA-AwS angestrebt.

CRM / Finanzwesen: Die zu erbringenden Aufgaben beider betrieblicher Objekte werden als relativ stabil angesehen. Die Analyse der AwS-Landschaft hat zudem ergeben, dass es bereits eine Kundenverwaltung und ein ERP-basiertes Finanzmodul im Unternehmen gibt. Beide Systeme werden bisher in anderen Kontexten im Unternehmen genutzt und decken über eine Teilfunktionalität die zu erbringenden Aufgaben der genannten betrieblichen Objekte ab, so dass eine Unterstützung von *CRM* und *Finanzwesen* weiterhin durch diese AwS erfolgen soll.

Flottenlenkung / Car-Pool / e-Car: Auch die Aufgaben dieser betrieblichen Objekte werden insgesamt als relativ stabil angesehen. Daher soll ein standardisiertes marktgängiges AwS zur Erfüllung der Aufgaben der genannten Objekte genutzt werden.

Eine Überprüfung der in Abb. C-75 dargestellten AwS-Kartierung zeigt, dass jedem betrieblichen Objekt genau ein AwS zugeordnet ist. Aus Sicht der SOM-Methodik ist somit keine Verfeinerung des GP-Modells erforderlich.

Schritt 3: Fachliche Spezifikation des SOA-AwS. Abb. C-76 zeigt das initiale KOS für das abgegrenzte SOA-AwS, welches gemäß der SOM-Methodik direkt aus dem GP-Modell abgeleitet wurde. Das initiale KOS zeigt bereits die benötigten konzeptuellen Objekttypen und die entsprechenden Abhängigkeiten zwischen diesen.

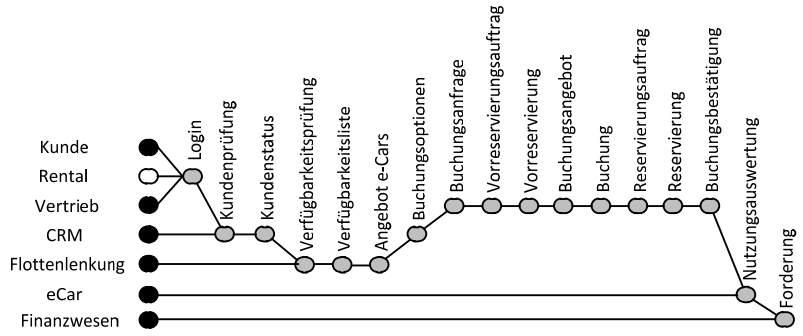


Abb. C-76: Initiales konzeptuelles Objektschema

Aus Datensicht befindet sich das gezeigte Schema allerdings noch nicht in wünschenswerter Form, so dass in einer weiteren Verfeinerung u. a. Aufgabenobjekte mit sich weitgehend überlappenden Attributen oder Operatoren im Hinblick auf Vermeidung von Daten- und Funktionsredundanz zusammengefasst werden müssen (Ferstl und Sinz 2008, S. 222f). Als Ergebnis dieser Zusammenfassung ergeben sich die KOTs *Mietvertrag*, *Kundenkategorie* und *Verfügbarkeitsliste / Verfügbarkeitsposition*. Die Unterscheidung der KOTs *Verfügbarkeitsliste* und *Verfügbarkeitsposition* erfolgt aus Gründen der Normalisierung, die genannten KOTs sind über eine is-part-Beziehungen verknüpft. Der existenzunabhängige KOT *Vertrieb* wird aus dem KOS entfernt, da hierfür keine spezifischen Attribute oder Operatoren identifiziert werden können. Abb. C-77 zeigt das überarbeitete und konsolidierte KOS.

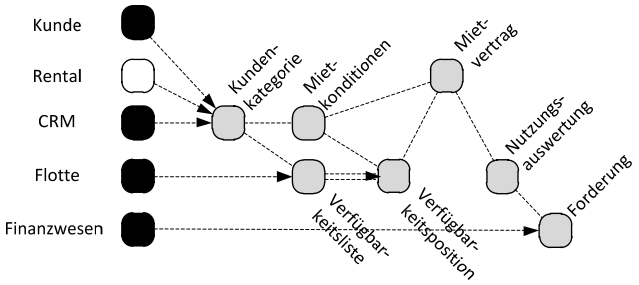


Abb. C-77: Konsolidiertes konzeptuelles Objektschema

Das in Abb. C-78 dargestellte VOS bezieht sich als Teil der Spezifikation für das SOA-AwS ausschließlich auf das betriebliche Objekt *Vertrieb*; es wurde gemäß der SOM-Methodik direkt aus dem zugehörigen VES abgeleitet. Würde die Abgrenzung des SOA-AwS mehrere betriebliche Objekte umfassen, so müssten auch diese entsprechend im VOS berücksichtigt werden. Im Unterschied zum KOS erfolgt im VOS keine weitere Konsolidierung, da bereits die initiale Grundstruktur eine hinreichende Basis für die weiteren Schritte des Ansatzes darstellt.

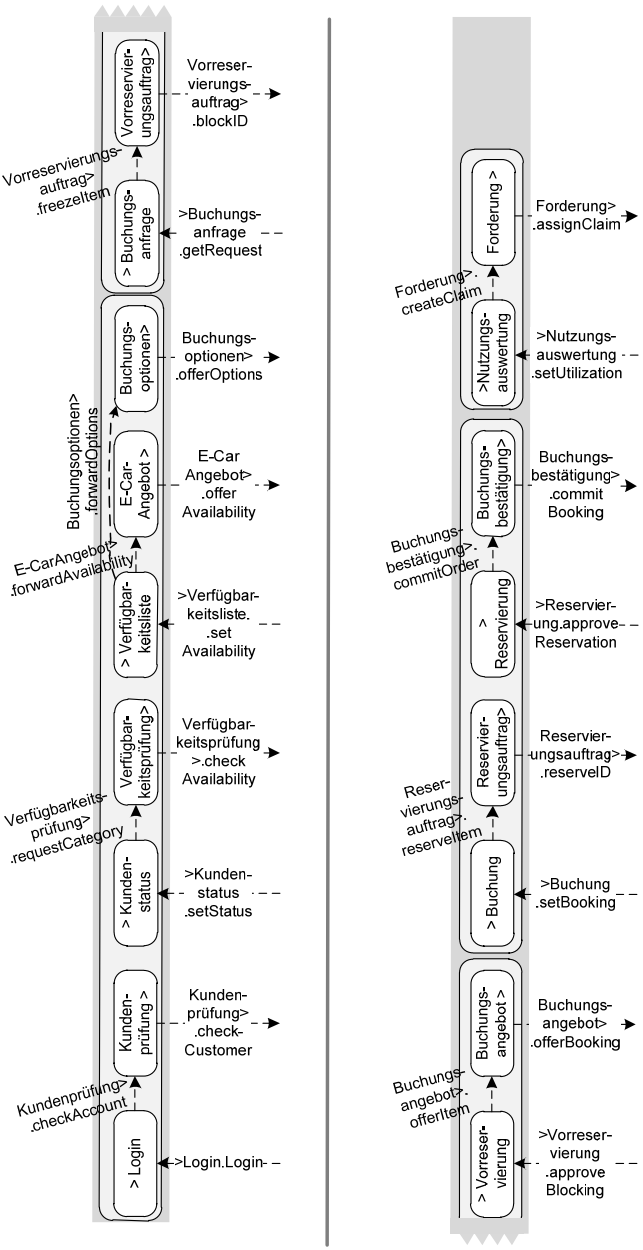


Abb. C-78: VOS für die Aufgaben des betrieblichen Objekts Vertrieb

Schritt 4: Externe Schnittstellen des SOA-AwS. Aus der Kartierung in Schritt 2 ergeben sich die erforderlichen Prozessschnittstellen zwischen dem SOA-AwS und den umgebenden AwS *CRM-AwS*, *FI-AwS* und *Flotten-AwS*. Die Spezifikation der Schnittstellen erfolgt in Form von Vorgangs-Services, die aus dem in Schritt 3 spezifizierten VOS abgeleitet werden. Dies soll stellvertretend für alle umgebenden AwS am Beispiel der Prozessschnittstelle zwischen dem *CRM-AwS* und dem *SOA-AwS* erläutert werden: Laut Kartierung weist das *CRM-AwS* eine bidirektionale Schnittstelle zum SOA-AwS auf. Im VOS finden sich die genannten Schnittstellen in Form von aus- und eingehenden Nachrichtenflüssen wieder. Konkret handelt es sich hierbei um die Nachrichtenflüsse *Kundenprüfung>.checkCustomer* und *>Kundenstatus.setStatus*. Auf Basis einer Ableitung der identifizierten Prozessschnittstellen und damit korrespondierenden Nachrichtenflüsse im VOS kann eine Abgrenzung der erforderlichen Vorgangs-Services erfolgen. Im Sinne des *In-Out-MEP* kann *Kundenprüfung>.checkCustomer* als Aufruf eines Vorgangs-Services interpretiert werden, während *>Kundenstatus.setStatus* die Servicerückmeldung hierzu darstellt. Dementsprechend werden die genannten Nachrichtenflüsse in einen elementaren Vorgangs-Service *Kundenauskunft-Service* überführt. Im Unterschied hierzu lässt sich für das *FI-AwS* lediglich ein unidirektionaler Nachrichtenfluss *Forderung.assignClaim* identifizieren, so dass in diesem Fall das *In-Only-MEP* anzuwenden ist und zu einem Serviceaufruf ohne Rückmeldung führt.

Zur Bestimmung erforderlicher Datenschnittstellen zwischen dem *CRM-AwS* und den damit über Nachrichtenflüsse verbundenen VOTs *Kundenprüfung>* und *>Kundenstatus* werden die Aufgabenobjekte herangezogen, auf denen die genannten VOTs operieren. Im Falle des VOT *>Kundenstatus* sind dies die KOTs *Kunde* und *Kundenkategorie*, wobei *Kundenkategorie* ein überlappendes Aufgabenobjekt darstellt, das durch den zuvor spezifizierten Vorgangs-Service an *>Kundenprüfung* übermittelt wird. Die Datenhoheit soll in diesem Fall dem *CRM-AwS* zugeschrieben werden. Eine Datenschnittstelle wäre nur dann erforderlich, wenn die Änderung der Kundenkategorie im *CRM-AwS* Auswirkung auf eine laufende Instanz des Geschäftsprozesses hätte.

Schritt 5: Serviceidentifizierung. Die Innensicht des SOA-AwS wird in Form von Entitäts- und Vorgangs-Services spezifiziert.

Entitäts-Services werden hierbei direkt aus dem KOS abgeleitet. Die für das e-Car Rental identifizierten Entitäts-Services sind in dem in Abb. C-79 nochmals aufgeführten konsolidierten KOS skizziert. In Hinblick auf eine angemessene Granularität der Services wurden einzelne KOTs zu einem gemeinsamen Enti-

täts-Service zusammengefasst. Im Falle des *Angebots-Services* z. B. wurden hierzu die KOTs *Verfügbarkeitsposition*, *Verfügbarkeitsliste* und *Mietkonditionen* zu einem darauf operierenden Entitäts-Service zusammengefasst.

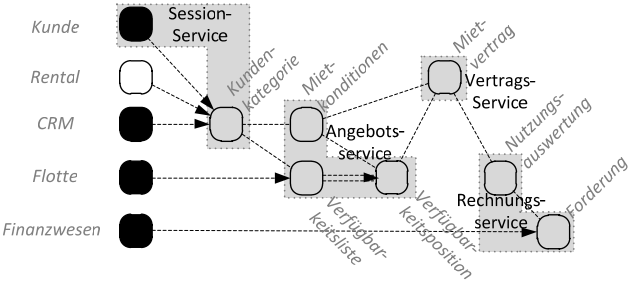


Abb. C-79: Entitäts-Services abgeleitet aus dem KOS

Die Abgrenzung der nicht-elementaren Vorgangs-Services ist im VOS (Abb. C-78) skizziert. So kann die VOT-Sequenz >*Login* bis *Buchungsoptionen*> aus Ablaufsicht als zusammenhängendes Vorgangsnetz aufgefasst werden, das in einen Vorgangs-Service *erstelleKatalog* überführt werden kann. Zu beachten ist hierbei, dass >*Login* und *Buchungsoptionen*> eine direkte Transaktionsbeziehung zum Kunden aufweisen und das abgegrenzte Vorgangsnetz im Sinne des *In-Out-MEP* zu interpretieren ist. Die Übertragung des *In-Out-MEP* auf einen Teilausschnitt des VOS kann daher eine nützliche Hilfestellung bei der Abgrenzung nicht-elementarer Vorgangs-Services sein.

Grundlage der nicht-elementaren Vorgangs-Services sind die einzelnen VOTs, die in Form elementarer Vorgangs-Services abgebildet werden.

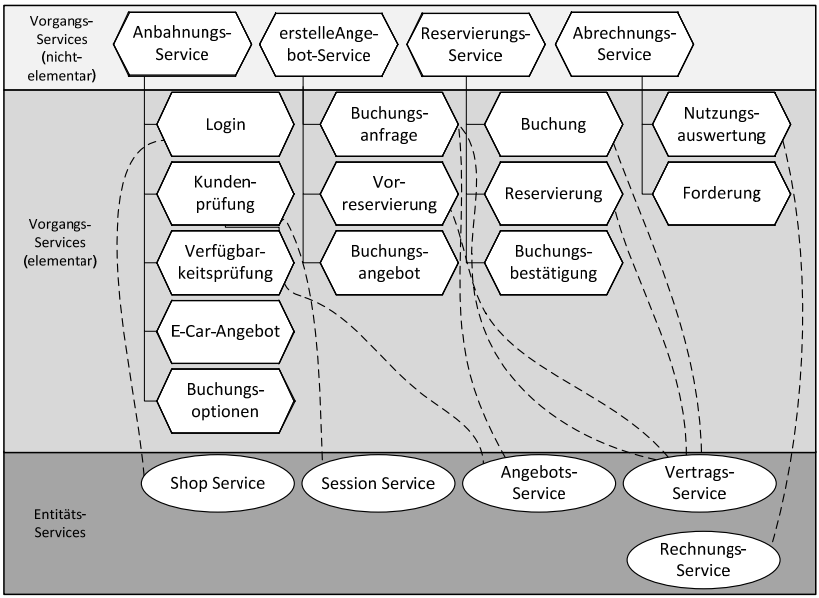


Abb. C-80: Service-orientierte Architektur des SOA-AwS

Abb. C-80 stellt die Innensicht des SOA-AwS in Form einer service-orientierten Architektur auf fachlicher Ebene zusammenfassend dar.

14.5 Zusammenfassung und Ausblick

Ziel des vorliegenden Beitrags ist die Entwicklung eines Ansatzes für den fachlichen Entwurf von SOA-AwS als Teilsysteme einer partiellen SOA der betrieblichen AwS-Landschaft. Die Entwicklung erfolgt modellbasiert auf der Basis eines GP-Modells. Motivation für den Ansatz ist die Annahme, dass in vielen Fällen nicht die vollständige AwS-Landschaft gemäß SOA gestaltet oder transformiert werden soll, sondern lediglich diejenigen Teile, welche Geschäftsprozesse mit hohen Flexibilitätsanforderungen unterstützen. Die Anwendung wurde an einem nicht-trivialen Fallbeispiel demonstriert.

Die Grundlage für den vorgestellten Ansatz bildet das Semantische Objektmodell (SOM) als umfassende Methodik für die Modellierung betrieblicher Systeme (Ferstl und Sinz 2008, S. 192ff). Das vorgeschlagene Vorgehen für den Entwurf einer partiellen SOA ist durchgängig modellbasiert. Ausgangspunkt ist ein SOM-GP-Modell, in dem ein Ausschnitt mit hohem Flexibilitätsbedarf identifiziert wurde.

Weiterer Forschungsbedarf besteht u. a. hinsichtlich der Durchführung weiterer Fallstudien, der Verfeinerung des modellbasierten Vorgehens zu einer modellgetriebenen Entwicklungsmethodik auf Basis von Metamodellen, der Werkzeugunterstützung sowie der softwaretechnischen Implementierung prototypischer SOA-AwS. Im derzeitigen Entwicklungsstand ausgeklammert sind die explizite Berücksichtigung teilautomatisierter Geschäftsprozesse und damit Fragen der Gestaltung von Nutzerschnittstellen.

14.6 Literatur

Aier S, Winter R (2009) Virtuelle Entkopplung von fachlichen und IT-Strukturen für das IT/Business Alignment - Grundlagen, Architekturgestaltung und Umsetzung am Beispiel der Domänenbildung. WIRTSCHAFTSINFORMATIK 51(2):175–191.

Alahmari S, Zaluska E, Roure D de (2010) A Service Identification Framework for Legacy System Migration into SOA. IEEE Computer Society (Hrsg.) IEEE Seventh International Conference:614–617.

Bell M (2010) SOA Modeling Patterns for Service-Oriented Discovery and Analysis. Wiley, Hoboken, NJ.

Berlecon Research (2006) SOA in der Praxis. Wie Unternehmen SOA erfolgreich einsetzen. http://www.berlecon.de/research/reports.php?we_objectID=271. Abruf am 2010-08-22.

Bernhard R, Jahn BU (2009) Modellgetriebene Entwicklung von serviceorientierten Architekturen. In: Hansen H, Karagiannis D, Fill H (Hrsg.) Business Services: Konzepte, Technologien, Anwendungen. 9. Internationale Tagung Wirtschaftsinformatik. Wien, 25. – 27. Februar 2009, Wien.

Carey MJ (2008) SOA What? IEEE Computer 41(3):92–94.

Eckert, Julian; Repp, Nicolas; Martin, Wolfgang (2010): SOA Check 2010. Status Quo und Trends im Vergleich zum SOA Check 2007 bis 2009. S.A.R.L. Martin/TU Darmstadt/IT Research. Online verfügbar unter www.soa-check.eu, zuletzt geprüft am 17.03.2011.

Erl T (2008) SOA. Principles of Service Design. Prentice Hall PTR, Upper Saddle River, NJ.

Erl T (2009) SOA. Design Patterns. Prentice Hall PTR, Upper Saddle River, NJ.

Ferstl OK, Sinz EJ (1995) Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. WIRTSCHAFTSINFORMATIK 37(3):209–220.

Ferstl OK, Sinz EJ (2008) Grundlagen der Wirtschaftsinformatik. 6. Auflage, Oldenbourg, München.

Hess A, Humm B, Voß M (2006) Regeln für serviceorientierte Architekturen hoher Qualität. Informatik Spektrum 29(6):395–411.

Heutschi R (2007) Serviceorientierte Architektur. Architekturprinzipien und Umsetzung in die Praxis. Springer-Verlag, Berlin, Heidelberg.

Hutchinson J, Gerald Kotonya, James Walkerdine, Peter Sawyer, Glen Dobson, Victor Onditi (2008) Migrating to SOAs by Way of Hybrid Systems. IT Professional 10(1):34–42.

Josuttis NM (2007) SOA in Practice. The Art of Distributed System Design. O'Reilly Media, Sebastopol, CA.

Kätker S, Patig S (2009) Model-driven Development of Serviceoriented Business Application Systems. In: Hansen H, Karagiannis D, Fill H (Hrsg.) Business Services: Konzepte, Technologien, Anwendungen. 9. Internationale Tagung Wirtschaftsinformatik. Wien, 25. – 27. Februar 2009, Wien.

Kavianpour M (2007): SOA and Large Scale and Complex Enterprise Transformation. In: Bernd J. Krämer, Kwei-Jay Lin und Priya Narasimhan (Hrsg.): Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 530–545.

Keller G, Nüttgens N, Scheer A (1992) Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK), Saarbrücken.

Krafzig D, Banke K, Slama D (2005) Enterprise SOA. Service-Oriented Architecture Best Practices. Prentice Hall PTR, Upper Saddle River, NJ.

Leyking K, Dreifus F, Loos P (2007) Serviceorientierte Architekturen. WIRTSCHAFTSINFORMATIK 49(5):394–401.

Masak D (2005) Moderne Enterprise Architekturen. Springer-Verlag, Berlin, Heidelberg.

Masak D (2007) SOA? Serviceorientierung in Business und Software. Springer-Verlag, Berlin, Heidelberg.

Melzer I (2010) Service-orientierte Architekturen mit Web Services, 4. Aufl., Spektrum Akademischer Verlag, Heidelberg, Neckar.

Object Management Group (OMG) (2009) Business Process Model and Notation (BPMN). Version 1.2. http://www.omg.org/spec/BPMN/1.2/PDF/. Abruf am 2010-08-22.

Papazoglou MP (2008) Web Services. Principles and Technology. Pearson, Harlow.

Peltz C (2003) Web Services Orchestration and Choreography. IEEE Computer 36(10):46–52.

Richter J, Haller H, Schrey P (2005) Serviceorientierte Architektur. Informatik Spektrum 28(5):413–416.

Schelp J, Winter R (2008) Entwurf von Anwendungssystemen und Entwurf von Enterprise Services - Ähnlichkeiten und Unterschiede. WIRTSCHAFTSINFORMATIK 50(1):6–15.

Schlimm N (2010) Serviceorientierte Architektur - eine Standortanalyse. Informatik Spektrum 33(3):282–287.

Siedersleben J (2007) SOA revisited: Komponentenorientierung bei Systemlandschaften. WIRTSCHAFTSINFORMATIK 49(Sonderheft):110–117.

Sinz EJ (2002) Architektur von Informationssystemen. In: Rechenberg P, Pomberger G (Hrsg.) Informatik-Handbuch. Hanser-Verlag, München.

Sinz, EJ (2008) SOA und die bewährten methodischen Grundlagen der Entwicklung betrieblicher IT-Systeme. In: WIRTSCHAFTSINFORMATIK 50 (1), S. 70–72.

Thomas O, Leyking K, Scheid M (2009) Vorgehensmodelle zur Entwicklung serviceorientierter Softwaresysteme. In: Hansen H, Karagiannis D, Fill H (Hrsg.) Business Services: Konzepte, Technologien, Anwendungen. 9. Internationale Tagung Wirtschaftsinformatik. Wien, 25. – 27. Februar 2009, Wien.

W3C (2007): Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts. W3C (World Wide Web Consortium). Online verfügbar unter http://www.w3.org/TR/wsdl20-adjuncts/, zuletzt geprüft am 16.03.2011.

Wagner, Heinz-Theo; Weitzel, Tim (2006): Operational IT Business Alignment as the Missing Link from IT Strategy to Firm Success. In: Proceedings of the Twelfth Americas Conference on Information Systems. AMCIS 2006. Acapulco, Mexico, S. 570–578.

Wilde T, Hess T (2007) Forschungsmethoden der Wirtschaftsinformatik. Eine empirische Untersuchung. WIRTSCHAFTSINFORMATIK 49(4):280–287.

Woods D, Mattern T (2006) Enterprise SOA: Designing IT for Business Innovation. O'Reilly, Beijing.