Die Adaptive Services Grid Plattform: Motivation, Potential, Funktionsweise und Anwendungsszenarien

Dominik Kuropka, Mathias Weske
Hasso-Plattner-Institut, Universität Potsdam
{dominik.kuropka | mathias.weske}@hpi.uni-potsdam.de

Abstract: Dienstbasierte Softwarearchitekturen werden zukünftig den Kern betrieblicher IT-Landschaften bilden. In diesem Beitrag wird zunächst der Stand der Technik im Bereich dienstbasierter Architekturen vorgestellt. Anhand eines konkreten Fallbeispiels wird aufgezeigt, welche zentralen Anforderungen durch diese Architekturen bislang nicht erfüllt werden. Die Autoren argumentieren, dass semantisch reiche Beschreibungen von Diensten notwendig sind, um das volle Potential dienstbasierter Architekturen ausschöpfen zu können, unter anderem dynamisches Auffinden und Binden von Diensten, die automatische Integration neuer Dienste sowie die kostengünstige Entwicklung von Mehrwertdiensten durch Komposition semantisch beschriebener Basisdienste. Im weiteren Verlauf des Beitrages wird die Adaptive Services Grid (ASG) Plattform vorgestellt, die dynamisches Auffinden und Binden von semantisch beschriebenen Diensten implementiert. Zum Abschluss werden ihre Funktionsweise und mögliche Anwendungsszenarien skizziert.

Einleitung

Heutige Softwarearchitekturen in betrieblichen Umgebungen stehen vor der Herausforderung, Anforderungen des Marktes schnell und kostengünstig erfüllen zu müssen. Weil durch schnelle und adäquate Reaktion auf neue Anforderung ein erheblicher Wettbewerbsvorteil realisiert werden kann, spielt die Entwicklung entsprechender Systeme und Technologien eine zentrale Rolle beim Entwurf von Softwarearchitekturen in betrieblichen Umgebungen. Eine zentrale Eigenschaft dieser Architekturen ist die Bereitstellung klar definierter Funktionalität, die auf modulare Weise verfügbar ist und entsprechend vielfältig eingesetzt werden kann. Dafür bilden dienstbasierte Softwarearchitekturen einen viel versprechenden Ansatz; man kann davon ausgehen, dass diese zukünftig den Kern betrieblicher IT-Landschaften bilden werden.

In diesem Beitrag wird zunächst der Stand der Technik im Bereich dienstbasierter Architekturen vorgestellt. Anhand eines konkreten Fallbeispiels wird aufgezeigt, welche zentralen Anforderungen durch diese Architekturen bislang nicht erfüllt werden. Die Autoren argumentieren, dass semantisch reiche Beschreibungen von Diensten notwendig sind, um das volle Potential dienstbasierter Architekturen ausschöpfen zu können. Dazu gehören unter anderem dynamisches Auffinden und Binden von Diensten, die automatische Integration neuer Dienste sowie die kostengünstige Entwicklung von Mehrwertdiensten durch Komposition semantisch beschriebener Basisdienste. Zu Abschluss des Beitrages wird die Adaptive Services Grid (ASG) Plattform vorgestellt. Diese implementiert dynamisches Auffinden und Binden von semantisch beschriebenen Diensten. Ihre Funktionsweise und insbesondere der Lebenszyklus der Diensterbringung wird diskutiert. Zusätzlich werden mögliche Anwendungsszenarien skizziert.

Stand der Technik

Der zunehmende Konkurrenzdruck in der Softwareindustrie zwingt Systemanbieter dazu, ihren Kundenkreis stetig zu vergrößern, um auf diese Weise Kostenreduktion durch Skaleneffekte bei der Software-Entwicklung und -wartung zu realisieren. Um mehr Kunden zu gewinnen ist es wichtig, die angebotene Funktionalität zu erhöhen bzw. existierende Funktionalität auf einfache Weise zugänglich zu machen. Diese Anforderungen haben aber im Allgemeinen auch eine Erhöhung der Komplexität und damit auch eine überproportionale Steigerung der Kosten für Softwareentwicklung und -wartung zur Folge, wenn dieser Tendenz nicht durch Strategien der Kom-

plexitätsreduktion begegnet wird. Eine auf technischer Ebene angesiedelte und in vielen Szenarios erfolgreiche Strategie ist die Aufteilung der Funktionalität in angemessene, wieder verwendbare und ggf. verteilbare Komponenten, die gemeinsam die Systemfunktionalität realisieren. Bekannte Technologien zur Umsetzung derartiger komponentenbasierter Systeme sind Remote Procedure Calls (RPC) [1], Common Object Request Broker Architecture (CORBA) [2] und Web Services [18, 19].

Lange Zeit ist die Aufteilung von Softwaresystemen in wieder verwendbare Komponenten von Softwareentwicklern fast ausschließlich nach technischen Gesichtspunkten durchgeführt worden. Dieses ist besonders dann angemessen, wenn die Teilkomponenten einer einzelnen Organisation angehören. Der Bedarf an kurzen Reaktionszeiten am Markt (flexible Anpassung ganzer Beschaffungs- und Produktionsketten auf kurzfristige Nachfrageschwankungen nach Endprodukten) erzwingt jedoch zunehmend auch die Integration von externen Informationssystemen und Teilbereichen externer IT-Systeme. Für derartige Integrationsanforderungen hat sich die klassische, rein nach technischen Wiederverwendbarkeitsaspekten orientierte Komponentenaufteilung als zu feingranular, zu sehr zueinander interdependent und damit als schwer integrierbar herausgestellt. Um diese Herausforderung in den Griff zu bekommen, sollten Softwarekomponenten nach fachlichen Gesichtspunkten Geschäftsfunktionalität in Form von dedizierten Diensten anbieten. Derartige Dienste werden sowohl im englischen als auch im deutschen Sprachgebrauch als Services bezeichnet.

Ein Service ist eine wohldefinierte Funktionalität im Sinne einer Dienstleistung, die als Reaktion auf eine elektronische Anfrage erbracht wird. Zur Anwendung von Services wird eine Service Orientierte Architektur (SOA) benötigt, die die verschiedenen Rollen der Beteiligten definiert [4]: Der Service Provider ist dabei diejenige Institution, die einen Service anbietet, indem Sie ihn bei einem Service Broker publiziert. Der Service Provider ist für die Ausführung des Dienstes verantwortlich und muss daher eine entsprechende Infrastruktur bereitzustellen. Der Service Broker bietet ein Verzeichnis an, in dem Services registriert werden können und in dem nach vorhandenen Diensten gesucht werden kann. Bei dem Service Broker kann es sich durchaus um dieselbe Institution handeln wie beim Service Provider oder Requestor. Der Service Requestor wiederum ist diejenige Institution, die einen Service anfragt. Dafür wendet sich der Service Requestor zunächst an den Service Broker, um einen geeigneten Service zu finden. Hat er diesen gefunden, erfährt er aus der Spezifikation des Services wer der Service Provider ist und wie er den Service technisch aufzurufen hat.

Für Anbieter und Nutzer von ERP-Software hat die Verwendung von Services zwei Vorteile: Komplexitätsreduktion und verbesserte Integrierbarkeit. Die Komplexitätsreduktion ergibt sich aus der Tatsache, dass Serviceorientierte Architekturen die betriebliche Gesamtfunktionalität eines Systems in einzelne gekapselte Teilfunktionalitäten aufteilen und einzeln verfügbar machen. Für die Adaption von ERP-Systemen an das Betriebsumfeld von Vorteil, dass die einzelnen Services nach betriebswirtschaftlichen und nicht vorwiegend nach technischen Gesichtspunkten aufgestellt, gekapselt und dokumentiert werden. Dadurch lassen sich die Services der ERP-Software mit bereits im Unternehmen vorhandenen bzw. noch zu implementierenden Services leichter zu durchgehenden Geschäftsprozessen komponieren und in verschiedenen Anwendungsystemen integrieren..

Auf technischer Seite können Services prinzipiell mit unterschiedlichen Technologien realisiert werden, die einen Aufruf von Komponentenfunktionalität über Netzwerke ermöglichen. Relativ weit verbreitet sind die standardisierten Web Services. Diese verwenden zumeist das etablierte und aus dem Umfeld des World Wide Web bekannte Hypertext Transfer Protocol (HTTP) [5] als Kommunikationsbasis. Als universelles Datenformat wird dabei die Extensible Markup Language (XML) [6] verwendet, welches sich durch seine Flexibilität und die weite Verfügbarkeit von Werkzeugen und Bibliotheken zur Verarbeitung von XML-Dokumenten auszeichnet. Der Abruf von Funktionalität, also der Service-Aufruf wird üblicher Weise über das Simple Object Access Protocol (SOAP) [7] abgewickelt. Neben den Aufruf- und Kommunikationsprotokollen sind für Web Services auch standardisierte Formate zur Beschreibung der Web Services und zum Auffinden von Web Services definiert. Web Services werden üblicher Weise mit Hilfe eines Dokumentes beschrieben, welches im Format der Web Services Description Language (WSDL) [8] vorliegt. Dieses Dokument beschreibt wie ein Service-Aufruf technisch umzusetzen ist. Es enthält somit Informationen an welchen Server der Aufruf abgesetzt werden soll, welcher Methodenname anzugeben ist und gemäß welchem XML-Schema die Eingabe- und Ausgabedaten (Nachrichten) kodiert werden. Diese WSDL-Dokumente werden bei der Registrierung von Services bei dem jeweiligen Service Broker hinterlegt. Auch für die Kommunikation mit dem Service Verzeichnissen ist für Web Services ein Standard vorgegeben: Universal Description, Discovery and Integration (UDDI) [9]. Dieser Standard sieht vor, dass

Services unter Anderem anhand von Geschäftskategorien und Servicetypen (tModels) spezifiziert und gesucht werden können. Zusätzlich dazu besteht die Möglichkeit natürlichsprachliche Service-Beschreibungen zu hinterlegen.

Auch wenn (Web) Services die Integration von Softwaresystemen durch ihre geschäftsorientierte Modularisierung erleichtern, sind sie nicht in der Lage, das grundsätzliche Problem der Integration zu lösen: Da ein Rechner nicht in der Lage ist die von Ihm verarbeiteten Daten im menschlichen Sinne zu „verstehen“ ist jedes Softwaresystem darauf angewiesen, dass die Verarbeitung der Daten bis ins kleinste Detail in Form eines Programms beschrieben wird. Dieses ist ein mühsamer Prozess, der qualifiziertes Personal benötigt, das nicht nur die technischen Aspekte, sondern auch die betriebswirtschaftliche Dimension der Daten und der dahinter liegenden Prozesse versteht. So ist es auch mit (Web) Services im Allgemeinen nicht ohne weiteres möglich, das Ergebnis eines Service-Aufrufs direkt an den nächsten Service weiterzugeben.

Ein konkretes aber dennoch einfaches Integrationsbeispiel für ein derartiges Problem wird im Folgenden kurz beschrieben: Bei der Anschaffung und Integration eines Customer Management Systems eines angelsächsischen Herstellers an das bereits existierende ERP-System sollen diese im Rahmen des nachfolgend beschriebenen Prozesses integriert werden. Bei einem eingehenden Anruf des für Auskünfte zuständigen Sachbearbeiters soll anhand der übermittelten Telefonnummer automatisch die wahrscheinlichste Adresse des anrufenden Kunden bestimmt werden und automatisch der offenen Auftrag dieses Kunden oder dieser Kundengruppe bei dem Sachbearbeiter auf dem Bildschirm erscheinen. Aus Gründen der Vereinfachung gehen wir davon aus, dass dieses Szenario mit dem Aufruf von lediglich zwei Diensten realisiert werden kann. Der erste Dienst wird vom Customer Management System zur Verfügung gestellt und bietet die Funktionalität anhand einer Telefonnummer die dazugehörige Adresse zu bestimmen. Da es sich um einen englischen Anbieter handelt, hat er als Ausgabe dieses Services für die Adresse das folgende Datenformat festgelegt: „full_name“, „address“, „city“, „state“, „ZIP“, „country“. Jeder einzelne dieser Adressblöcke enthält jeweils die Information folgenden Informationsbestandteile der Adresse: vollständiger Name, Anschrift, Stadt, Bundesland, PLZ, Land.

Der zweite für diesen Prozess benötigte Service wird von dem existierenden ERP-System bereitgestellt, dieser Service nimmt eine Adresse entgegen und bietet die Funktionalität eine Bildschirmmaske auf dem Arbeitsplatz des zuständigen Sachbearbeiters zu öffnen die alle offenen Aufträge zu der übergebenen Adresse zur Auswahl stellt. Da es sich hierbei um ein in Deutschland entwickeltes System handelt, nehmen wir an, dass die Adresse im folgenden Format an den Service zu übermitteln ist: „Name“, „Straße“, „Hausnummer“, „PLZ“, „Stadt“, „Land“. Wie man erkennt, passen diese Datenformate nicht zueinander. Ein gängiges Softwaresystem ist aus diesem Grund nicht in der Lage, die Daten von dem ersten Adressformat in das zweite zu transformieren, ohne dass ein Programmierer die entsprechenden Transformationsregeln vorgibt. Die Implementierung dieser Transformationsregeln ist selbst in diesem einfachen Beispiel relativ mühsam. So muss zum Beispiel aus der dem Block „address“ die Straße und Hausnummer durch geeignete Regeln extrahiert werden. Eine einfache eins-zu-eins Transformation, bei der lediglich die Blocknamen ausgetauscht werden, ist gerade bei komplexeren Datenstrukturen, wie zum Beispiel ganzen Rechnungen oder Aufträgen im Allgemeinen nicht möglich. Zusätzlich kommt noch hinzu, dass eine derartige Transformationsvorschrift unter ungünstigen Umständen für jede beliebige Verknüpfung von Services notwendig ist.

Neben dem Problem der Datentransformation ist auch die Komposition von Services zu Geschäftsprozessen eine schwierige Aufgabe. Die Komposition von Services ist deshalb von Bedeutung, weil sie die Wiederverwendung von Diensten massiv verbessert, und somit zur Kostenreduktion beiträgt. Jedoch wird für die Komposition von Diensten qualifiziertes Personal benötigt, welches in der Lage ist anhand einer Prozessspezifikation oder Aufgabenbeschreibung die geeigneten Services zu finden und diese korrekt, unter Berücksichtigung der ggf. jeweils notwendigen Transformationsregeln miteinander zu verbinden. Aufgrund der Tatsache, dass moderne Softwaresysteme durchaus sehr viele Services zur Verfügung stellen können und dass bei einer Integration mehrere Systeme beteiligt sind, die von verschiedenen Gruppen entwickelt wurden, ist das Auffinden der geeigneten Dienste lediglich anhand von einfachen, natürlichsprachlichen Klassifikationsmerkmalen – wie sie zum Beispiel von einem UDDI Verzeichnis zur Verfügung gestellt werden – ein schwieriges Unterfangen. Dieses ist unter anderem durch die unterschiedlichen Fachvokabulare und unterschiedlichen Interpretationen der Gegebenheiten bedingt. Durch den hohen manuellen Aufwand die eine Umsetzung eines Geschäftsprozesses mit sich bringt, ist eine stete Anpassung der Prozesse nur bedingt möglich. Daher besteht auch bei (Web) Services durchaus der

Bedarf, Maßnahmen zur Steigerung der Flexibilität und Implementierungseffizienz zu entwickeln und in zukünftigen Service-Plattformen umzusetzen.

Semantische Beschreibung von Services

Die Ursache für den hohen manuellen Aufwand bei der Verknüpfung von Services zu Geschäftsprozessen liegt darin, dass die formalen Service- und Datenschematabeschreibungen lediglich syntaktischer Natur sind. Das heißt, lediglich der technische Aufruf von Services und der Aufbau der dazugehörigen Datenstrukturen sind beschrieben. Eine formale Beschreibung der Funktionalität eines Services und der Bedeutung der verwendeten Datenstrukturen fehlt. Die ggf. vorliegende informale Beschreibung in natürlicher Sprache ist für einen Rechner weitgehend nutzlos. Auch ggf. vorliegende sprechende Namen für Methodennamen, einzelne Datenblöcke oder Attribute sind für einen Rechner nicht von Nutzen. Es macht für einen Rechner keinen Unterschied ob eine Datenstruktur als eine Sequenz von einzelnen Datenblöcken mit den sprechenden Benennungen „Name“, „Straße“, „Hausnummer“, „PLZ“, „Stadt“, „Land“ oder mit den abstrakten Benennungen „a“, „b“, c“, „d“, „e“, „f“ bezeichnet wurde. Der Rechner kann anhand der syntaktischen Strukturbeschreibung lediglich prüfen ob bei einer übergebenen Nachricht die entsprechenden Datenblöcke mit dem geforderten Namen vorhanden sind und er verfährt mit ihnen entsprechend seiner Programmierung.

Um ein Softwaresystem in die Lage zu versetzen, die Funktionalität und Bedeutung von Services und Datenstrukturen in begrenztem Ausmaß zu erfassen, ist es notwendig, die bisher rein syntaktischen Beschreibungen von Services und Definitionen von Datenstrukturen um semantische Beschreibungen zu erweitern. Eine Möglichkeit dieses zu erreichen, ist die Verwendung von logischen Ausdrücken die aus Konzepten einer bestehenden und akzeptierten Ontologie zusammengesetzt sind. Ursprünglich ist der Begriff der Ontologie von der Philosophie geprägt worden. Die Informatik hat ihn aufgegriffen, aber seine Bedeutung dabei an ihre Bedürfnisse angepasst [10, 11]. Während in der Informatik Einigkeit bezüglich der grundsätzlichen Bedeutung des Begriffes herrscht, sind Detailfragen noch in Diskussion. Deshalb hat sich noch keine allgemein akzeptierte Definition durchgesetzt. Diesem Artikel liegt die folgende Definition zu Grunde: Eine Ontologie ist ein Modell von sprachlichen Ausdrucksmitteln, auf die sich mehrere Akteure geeinigt haben und die für eine Kommunikation zwischen den Akteuren benutzt werden [12]. Ein Modell ist hierbei die Repräsentation eines Objektsystems für Zwecke eines Subjekts. Es entsteht durch die Konstruktionsleistung eines Menschen (Modellierer) und ist durch ein System von Symbolen (Modellierungssprache) dargestellt [13]. Eine Ontologie enthält somit Konzepte, Beziehungen zwischen diesen Konzepten und ihre jeweiligen Bezeichnungen die in einem bestimmten Umfeld von Kommunikationspartnern oder Sprachkreis eine feststehende und von allen Beteiligten anerkannte Bedeutung haben. In Unternehmen werden Ontologien gelegentlich unter der Bezeichnung Fachbegriffsmodelle verwendet. Für die Verwendung einer Ontologie in einem Softwaresystem ist es jedoch erforderlich, dass eine Ontologie in einer formalen Sprache beschrieben ist. Zurzeit gängige, teilweise noch in Entwicklung befindliche Sprachen sind im europäischen Raum die Web Service Modeling Language (WSML) [14] und im amerikanischen die Web Ontology Language (OWL) [15].

Die Adaptive Services Grid Plattform

Im Rahmen des von der Europäischen Union geförderten Adaptive Services Grid (ASG) [http://asg-platform.org] Projektes wird an einer Service-Plattform gearbeitet, bei der Datenstrukturen und Services semantisch spezifiziert werden und somit Services automatisiert auffindbar und zu Teilprozessen komponierbar sind. Diese Plattform ermöglicht die Spezifikation von Konzepten und Beziehungen zwischen diesen Konzepten einer Domäne in Form von einer Domänenontologie geschrieben in WSML. Diese dienen als Ausgangspunkt für die Spezifikation der Semantik von Datenschematas die ggf. unterschiedlichen Services zu Grunde liegen. Innerhalb der ASG-Plattform werden Daten im XML-Format weitergeleitet und verarbeitet, wobei das XML-Schema dieser Daten der Domänenontologie zugrunde liegt. Eine eindeutige, automatisierte Transformationsregel ermöglicht die Transformation der Ontologie in ein spezifisches XML-Schema. Somit wird über die Ontologie innerhalb der ASG-Plattform einer Art lokaler Standard für den Datenaustausch zwischen den Services geschaffen. Abbildung 1 zeigt wie ein Ausschnitt aus einer Domänenontologie zu dem obigen Integrationsbeispiel vereinfacht und visualisiert aussehen kann und wie die Datenstrukturen der Ontologie in ein XML-Schema (ebenfalls grafisch vereinfacht) dargestellt werden können. Des Weiteren zeigt diese Abbildung ebenfalls, wie die Daten-

strukturen des CRM bzw. ERP-Systems sich auf das Schema der ASG-Plattform abbilden lassen (Schema Mapping).

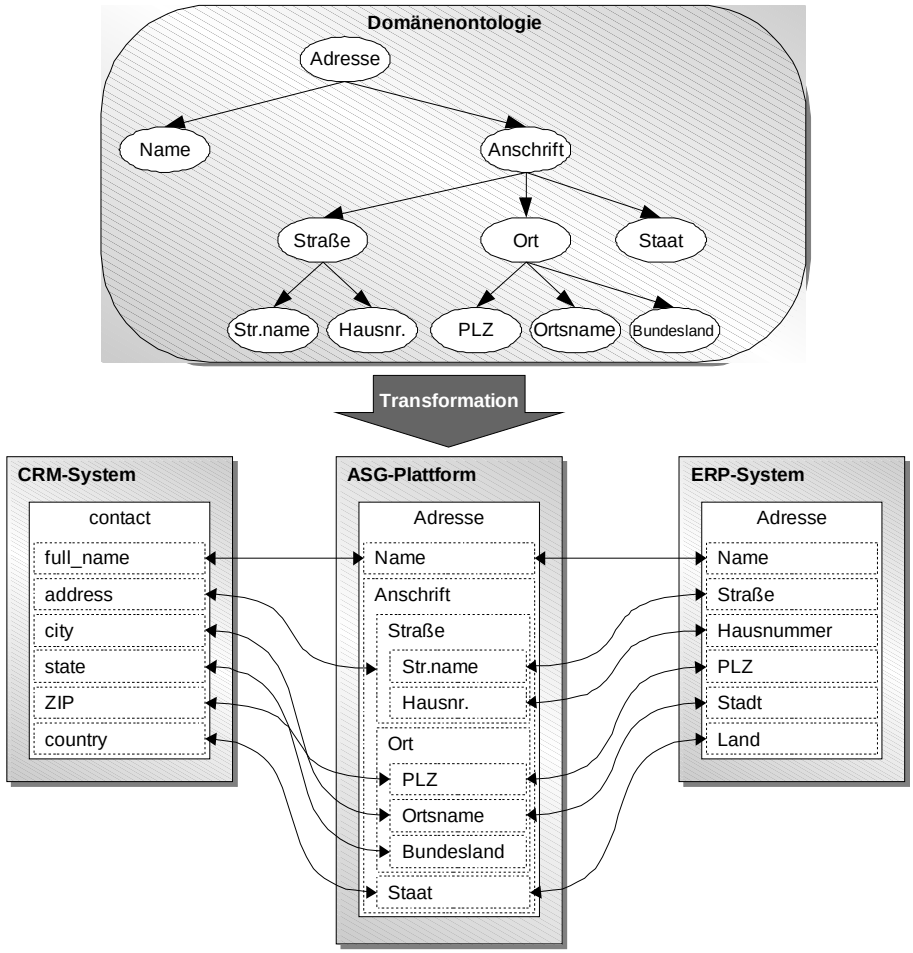

Abbildung 1: Mapping von Ontologie und Datenstrukturen

Die Verwendung eines zentralen, aus der Domänenontologie abgeleiteten Datenschemas hat den Vorteil, dass die Zahl der notwendigen Abbildungen von Schemas mit der Anzahl der zu integrierenden Systeme lediglich proportional wächst. Wird ein derartiges zentrales Datenschema nicht verwendet, dann wächst die Zahl der zu implementierenden Abbildungen bei der Integration quadratisch in der Anzahl der zu integrierenden Systeme bzw. Schnittstellen. Vergleiche dazu auch Abbildung 2.

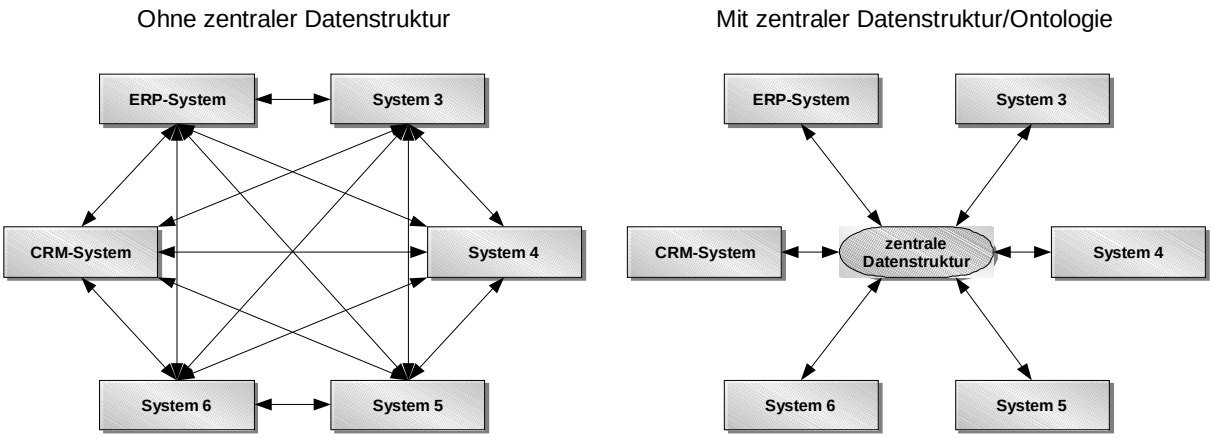

Abbildung 2: Mapping von Datenstrukturen ohne bzw. mit zentraler Datenstruktur

Damit ein Service innerhalb der ASG-Plattform verwendbar ist, muss er zunächst registriert werden. Bei der Registrierung werden folgende Informationen zu einem Service hinterlegt: die semantische Service Spezifikation, nicht-funktionale Eigenschaften des Service und das so genannte Service-Grounding. Die semantische Servi-

ce-Spezifikation definiert die Eingabe- und Ausgabedatenstrukturen des Dienstes unter Rückgriff auf die Domänenontologie. Zusätzlich können in der semantischen Service Spezifikation Vorbedingungen und Effekte (die über die einfache Datenein- und -ausgabe hinausgehen) definiert werden. So ist es zum Beispiel möglich, bei einem Service zu spezifizieren, dass er nicht nur eine Telefonnummer als Eingabe und eine Adresse als Ausgabe liefert, sondern dass die zurück gelieferte Adresse nicht irgendeine beliebige sondern die Adresse für eine eingegebenen Telefonnummer ist. (Im Beispiel wurde angenommen, dass jeder Telefonnummer genau eine Adresse zugeordnet ist.) Diese Information ist für eine vollständige semantische Spezifikation eines Services notwendig, da ansonsten im obigen Beispiel nicht klar ist, welcher Bezug zwischen Eingabe- und Ausgabedaten besteht. So könnte es ebenso einen weiteren Service geben, der ebenfalls eine Telefonnummer als Eingabe und eine Adresse als Ausgabe liefert, diese Adresse aber die Adresse des Telefonanbieters der jeweiligen Telefonnummer ist. Ohne einer vollständigen semantischen Spezifikation des Dienstes mit Vorbedingungen und Effekten ist eine derartige Unterscheidung der Funktionalität nur anhand von natürlichsprachlicher, und damit nicht automatisiert verarbeitbarer, Dokumentation möglich. Abbildung 3 zeigt in vereinfachter und visualisierter Form wie die Services aus dem Integrationsbeispiel (Service 1 und 2) semantisch spezifiziert werden können. Service 1 ist spezifiziert als ein Service der als Eingabe eine Telefonnummer erhält. Als Ausgabe liefert der Service eine Adresse, die der eingegebenen Telefonnummer zugeordnet ist. Zusätzlich wird sichergestellt, dass von diesem Service lediglich Kundenadressen zurückgeliefert werden. Service 2 hingegen benötigt eine Adresse eines Kunden (Kundenadresse) als Eingabe und liefert einen Auftrag als Rückgabe wobei dieser direkt auf dem Bildschirm des Sachbearbeiters angezeigt wird. Abschließend ist zu erwähnen, dass Service 3 zwar syntaktisch denselben Typ an Ein- und Ausgabe liefert wie Service 1. Jedoch liefert Service 3 im Unterschied zu Service 1 nicht die Adresse eines Kunden sondern die Adresse eines Telefonanbieters, der die eingegebene Telefonnummer bereitstellt. Ein solcher Unterschied in der Funktionalität ist auf der Ebene der reinen syntaktischen Definition eines Services nicht sichtbar, kann aber in einer formalen, semantischen Definition dargestellt und unterschieden werden.

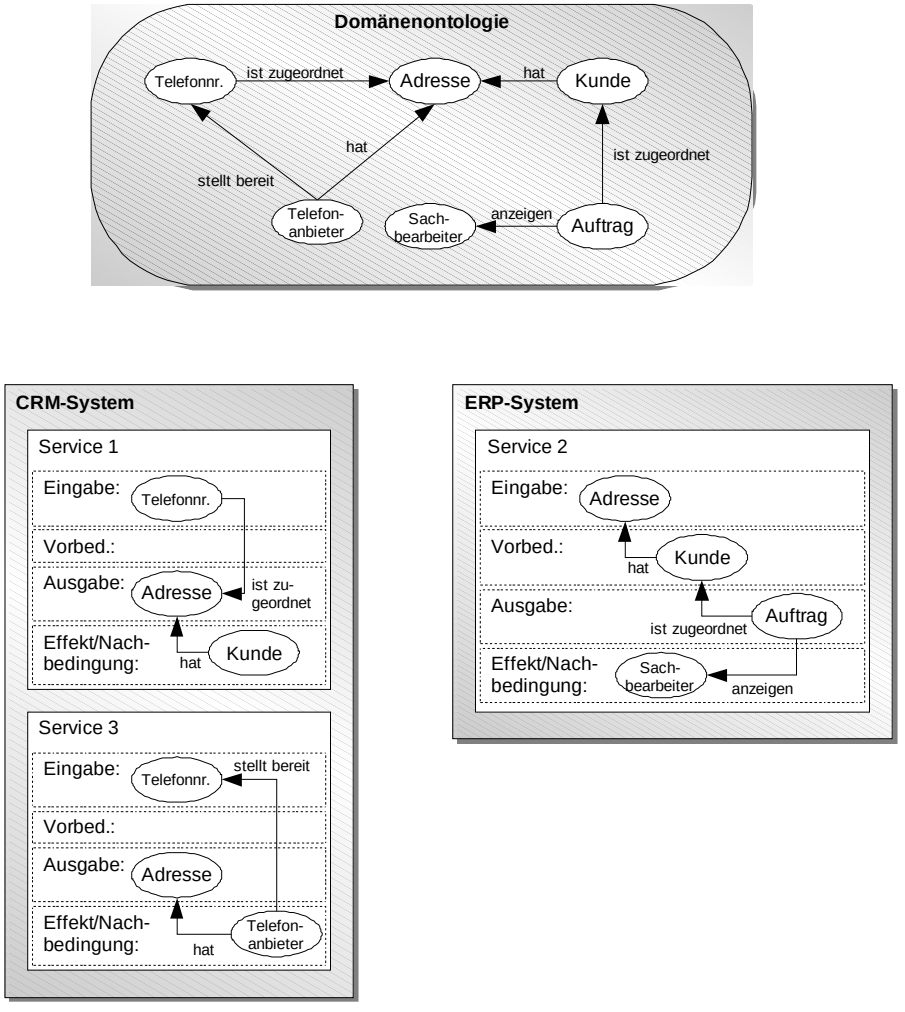

Abbildung 3: Semantische Spezifikation von Services

Die nicht-funktionalen Eigenschaften des Services enthalten statische und ggf. dynamisch zu bestimmende Informationen über den Dienst, die die Funktionalität des Dienstes nicht direkt aber ggf. indirekt betreffen. Statische Informationen können neben trivialen Daten wie einem arbiträren Service-Namen und dem Namen des Service-Providers auch Informationen über die maximal zu erwartende Bearbeitungszeit des Services oder die Kosten des Serviceaufrufs sein. Da diese jedoch nicht-funktional sind, dürfen die spezifizierten Eigenschaften nicht von den übergebenen oder resultierenden Daten selbst abhängen. Somit kann festgelegt werden, dass der Service „SMS-Versenden“ zum Beispiel immer 19 Cent kostet, unabhängig von der Telefonnummer. Soll der Preis des SMS-Versandes hingegen von den Daten und damit der Funktionalität des Services abhängen, dann müssen sie in der semantischen Service Spezifikation festgelegt werden. Neben einer rein statischen Spezifikation von nicht-funktionalen Eigenschaften erlaubt die ASG-Plattform auch die Spezifikation von dynamisch, zur Laufzeit zu bestimmenden nicht-funktionalen Eigenschaften: So kann beispielsweise Festgelegt werden, dass der Preis für den Versandt einer SMS zwischen 10 und 20 Cent kostet. Zur Laufzeit wird dann von der ASG-Plattform der Preis erfragt. Somit ist es zum Beispiel möglich für einen Service-Provider den Preis in Abhängigkeit von der momentanen Service-Auslastung zu spezifizieren. Im Falle, dass mehrere Service-Provider funktional gleiche Dienste wie zum Beispiel „SMS-Versenden“ anbieten, kann die ASG-Plattform zur Laufzeit die Preise erfragen und sich für den mit dem niedrigsten, weil zur Zeit am wenigsten ausgelasteten Service entscheiden.

Die technische Integration des Services wird über das Service-Grounding realisiert. Beim Service-Grounding wird aus der syntaktischen WSDL-Spezifikation eines Services mit Hilfe von teilautomatisierten Werkzeugen ein so genannter Service-Proxy erstellt. Die Aufgabe des Service-Proxies ist es den Service mit der ASG-Plattform zu integrieren. Der Service-Proxy bietet der ASG-Plattform ein standardisiertes Interface über das dynamische nicht-funktionale Eigenschaften des Service abgefragt und der Service selbst aufgerufen wird. Zusätzlich ist der Service-Proxy für die Konvertierung der Eingabe- und Ausgabedaten verantwortlich. So ist es zum Beispiel unter Verwendung von Extensible Stylesheet Language Transformation (XSLT) [16] möglich das vom Service verwendete XML-Datenschema an das durch die Ontologie vorgegebene XML-Schema innerhalb der ASG-Plattform zu transformieren. Somit können zur Laufzeit die Daten automatisiert in die jeweils benötigten Datenschemas konvertiert werden. Diese Transformationsregeln stellen somit auch gleichzeitig die semantische Spezifikation der vom Service verwendeten Datenschematas dar, weil diese das Service-Datenschema letztendlich über den Umweg des in der ASG Plattform verwendeten XML-Schemas auf die Domänenontologie abbilden.

Lebenszyklus der Serviceerbringung

Die Bearbeitung eines Aufrufs und die Bereitstellung eines Services durch die ASG-Plattform unterscheidet sich von dem Vorgehen heutiger Service Plattformen. Bei den heute gängigen Systemen wird ein bestimmter Service-Aufruf immer an einen statisch vorab gebundenen Service weitergeleitet. Bei der ASG-Plattform hingegen wird der Aufruf dynamisch an einen Service ausgeliefert oder bei Bedarf werden mehrere Services die eine Teilfunktionalität liefern können automatisiert zu einem neuen Service komponiert. Daher besteht der Aufruf eines Service bei der ASG-Plattform aus mehreren teilweise semantischen Bausteinen: Ausgangszustand, Zielzustand, sowie ggf. Nebenbedingungen für nicht-funktionale Eigenschaften und Optimierungskriterien. Von diesen sind die beiden Bausteine Ausgangszustand und Zielzustand die wichtigsten, weil diese die semantische Funktionalität der gewünschten Dienstleistung festlegen. Der Ausgangszustand umfasst eine semantische Beschreibung der Ist-Situation die als gegebene Ausgangsbasis für einen Service dient. Dazu gehören insbesondere die im Kontext der gewünschten Dienstleistung relevanten und bereits gegebenen Daten die in die Konzepte der Domänenontologie eingeordnet sein müssen. Der Zielzustand beschreibt das gewünschte Ergebnis des Service-Aufrufs welches ausgehend von dem gegebenen Ausgangszustand erreicht werden soll. Er besteht üblicher Weise aus der Auflistung der gewünschten Datenstrukturen bzw. Konzepten der Ontologie und Effekte. Der Ausgangszustand und der Zielzustand sind somit ähnlich zu der semantischen Spezifikation eines Services. Allerdings repräsentieren Sie nicht notwendigerweise die Funktionalität eines existierenden, sondern die Funktionalität eines gewünschten Services. Zusätzlich zu der gewünschten Funktionalität lassen sich Einschränkungen auf nicht-funktionalen Eigenschaften spezifizieren. Diese Einschränkungen können zum Beispiel genutzt werden, um beispielsweise aus Sicherheitsgründen sicher zu stellen oder auszuschließen, dass Services bestimmter Anbieter von der ASG-

Plattform zur Service-Erbringung ausgewählt werden. Alternativ können auch Restriktionen an beliebige andere, in der Domäne modellierte Eigenschaften von Services wie zum Beispiel Bearbeitungsdauer oder Preis definiert werden. Die Optimierungskriterien (z. B. minimale Bearbeitungsdauer oder kleinster Preis) dienen abschließend dazu der ASG-Plattform mitzuteilen für welche Services oder Service-Kompositionen sich die Plattform entscheiden soll, falls mehrere verschiedene, aber funktional ähnliche bzw. passende Services oder Service-Kompositionen zur Auswahl stehen.

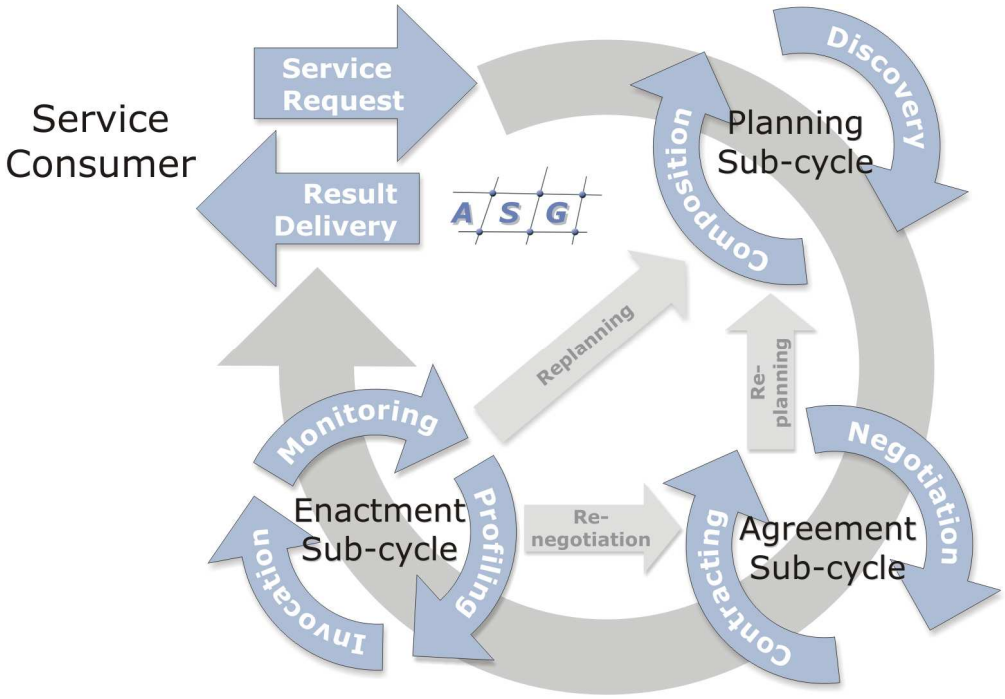

Abbildung 4: ASG Service Delivery Lifecycle

Eingehende Service-Anfragen werden von der ASG-Plattform in drei Schritten bearbeitet, siehe Abbildung 4. Im Ersten Schritt, dem Planning Sub-cycle wird im einfachsten Fall ein die Anfrage vollständig erfüllender Service gesucht. Kann ein derartiger Service nicht gefunden werden, dann versucht das System ausgehend von dem spezifizierten Ausgangszustand eine Service-Komposition zu finden, die zum Zielzustand führt. Dabei geht die ASG-Plattform schrittweise vor und identifiziert zunächst Services die vom Ausgangszustand aus ausführbar sind, weil deren Vorbedingung erfüllt ist. Für jeden dieser Services berechnet die Plattform den Zustand der nach Ausführung des Services zu erwarten ist. Dieser Vorgang wird als Virtual-Execution bezeichnet, weil die Services nicht wirklich ausgeführt werden, sondern lediglich anhand ihrer semantischen Spezifikationen eine Schätzung des Zustandes nach der Ausführung vorgenommen wird. Anhand dieses neuen Zustandes schätzt die Plattform ab in wie weit dieser neue Zustand näher am spezifizierten Zielzustand ist. Den in diesem Zusammenhang besten Service wählt die Plattform aus und fügt ihn zu einer vorläufigen Service-Komposition zusammen. Ausgehend von dem geschätzten Endzustand wird jetzt von neuem nach Services gesucht die ausführbar sind und deren Endzustand sich dem gewünschten Ziel annähert. Dieses geschieht so lange, bis der Zielzustand erreicht ist.

Die Service-Komposition wird innerhalb der Plattform als erweitertes BPEL4WS-Dokument [17] gespeichert. Die Erweiterung des Dokumentenformates ermöglicht die Speicherung von semantischen Beschreibungen für die einzelnen Services anstelle von fest gebundenen Services. Dadurch wird die Wiederverwendbarkeit und Flexibilität von Service-Kompositionen erhöht, da zur Ausführungszeit der Service-Komposition erst entschieden wird welcher konkrete Service ausgeführt werden soll. Somit ist es möglich die Entscheidung über einen konkreten Service immer noch dynamisch und flexibel zu gestalten aber gleichzeitig ein Komponierte oder per Hand vorgegebene Service-Kompositionen wieder zu verwenden ohne jedes Mal den Planning Sub-cycle vollständig durchlaufen zu müssen. Dieses ist insbesondere für häufig wiederkehrende Anfragen von Vorteil.

Im nächsten Schritt, dem Agreement Sub-cycle werden zu den semantischen Beschreibungen der Services der Service-Komposition aus dem ersten Schritt die passenden Services gesucht und an die Komposition gebunden.

Während in dem Planning Sub-cycle lediglich statische nicht-funktionale Eigenschaften berücksichtigt werden konnten, werden nun auch die dynamischen mit in die Betrachtung gezogen. Während des Agreement Sub-cycles werden zunächst zu jeder in der Service-Komposition enthaltenen Service-Spezifikation die funktional geeigneten Services herausgesucht. Anschließend werden dynamische nicht-funktionale Eigenschaften der Services erfragt bzw. ausgehandelt wenn dieses vom dem Service unterstützt wird. Anhand dieser Informationen wird während des Planning Sub-cycles eine solche Kombination von Services ausgewählt, die auf der einen Seite eventuell definierte Restriktionen an die nicht-funktionalen Eigenschaften einhält und auf der anderen Seite auch das Optimierungskriterium möglichst gut erfüllt. Diese ausgewählten Services werden dann anschließend an die Service-Komposition gebunden, die nun als normales BPEL4WS-Dokument im System vorliegt. Im Falle, dass funktionale Eigenschaften eines einzelnen Services ausgehandelt wurden, wird ein Service-Level-Agreement von beiden Seiten aufgesetzt und in der ASG-Plattform gespeichert.

Wie beim Planning-Subcycle besteht auch hier die Möglichkeit für häufige, zeitkritische Anfragen gebundene Service-Kompositionen zwischenzuspeichern oder manuell vorzugeben falls dieses erforderlich sein sollte.

Im normalerweise letzten Schritt, dem Enactment Sub-cycle werden die nun gebundenen Service-Kompositionen ausgeführt. Das heißt, dass die einzelnen Services entweder der Reihe nach oder wo möglich parallel ausgeführt werden. Der Aufruf der einzelnen Service wird dabei über den Service-Proxy durchgeführt, der die Daten gegebenenfalls erst in das benötigte XML-Datenschema transformiert und die Ergebnisse entsprechend in das aus der Domänenontologie abgeleitete Datenschema zurück transformiert. Nachdem alle Services der Komposition ausgeführt wurden ist im Normalfall das Ziel der Anfrage erreicht und das Ergebnis wird an den den Service Aufrufenden Service-Consumer zurück gegeben.

Da Services aus verschiedenen Gründen nicht immer erfolgreich ausgeführt werden können, implementiert die ASG-Plattform eine dreistufige Behandlung von Fehlern bzw. Ausnahmezuständen (Exception-Handling). Die erste Stufe ist innerhalb des Enactment Sub-cycles eingebettet. Mit ihr können kleinere Kommunikationsfehler vom Typus „Verbindung abgebrochen“ oder „Server nicht verfügbar“ korrigiert werden, sofern sich diese durch eine Reinitiierung der Kommunikationsverbindung oder Auswahl eines alternativen Servers bei dem betroffenen Provider lösen lassen. Eine derartige Korrektur beeinflusst weder nicht nicht-funktionalen Eigenschaften noch den Aufbau der semantischen Service-Komposition des Aufrufs als Ganzes. Fälle die nicht durch die erste Stufe behandelt werden können, z. B. weil aufgrund eines andauernden Netzausfalls keine Verbindung zu einem Server eines bestimmten Providers aufgebaut werden kann, werden an die zweite Stufe der Fehlerbehandlung weitergeleitet.

Diese zweite Stufe wird von dem Agreement Sub-Cycle implementiert und in ihr wird für einen konkreten semantisch spezifizierten Service-Baustein der Service-Komposition ein neuer, alternativer Provider gesucht. Kann ein solcher Provider gefunden werden, dann wird dieser an die Service-Komposition gebunden und die Ausführung wird wieder an den Enactment Sub-Cycle zur Weiterbearbeitung gegeben. Dadurch, dass ein neuer Provider ausgewählt wird, werden die nicht-funktionalen Eigenschaften der gesamten Service-Komposition geändert, wobei die grundsätzliche Struktur der Komposition jedoch erhalten bleibt. Neben der nicht-funktionalen Eigenschaft Provider können sich auch andere nicht-funktionale Eigenschaften wie z. B. der Preis oder die Bearbeitungsdauer der Serviceausführung ändern. Allerdings nur soweit wie die in der Ursprünglichen Anfrage definierten Restriktionen an die nicht-funktionalen Eigenschaften eingehalten werden. Daher besteht auch hier die Möglichkeit, dass der Fehler nicht behandelt werden kann, weil entweder kein Service-Provider den fehlerhaften Service substituieren kann oder weil die Bedingungen zu denen eine Substitution möglich ist gegen die definierten Restriktionen verstoßen. In diesem Fall wird der Fehler zum Neuplanen der gesamten Service-Komposition weiter an den Planning Sub-cycle gegeben.

In diesem dritten Schritt wird versucht den bisher erreichten Zustand der Service-Komposition als gegebenen Startzustand zu nehmen und von diesem Zustand aus das alte Ziel zu erreichen, ohne jedoch den Fehlerhaften Service zu verwenden. Diesem Ansatz liegt die Annahme zu Grunde, dass sich der fehlerhafte Service durch eine Komposition von mehreren kleineren Teilservices substituieren lässt oder durch die Verwendung von einen gänzlich Anderen Vorgehen vollständig eliminieren werden kann. Wird eine derartige Komposition gefunden, dann wird diese neue Komposition bestehend aus semantischen Service-Spezifikationen wie gewohnt zur weiteren Bearbeitung an den Agreement Sub-Cycle gegeben. Bei diesem Vorgehen ändern sich üblicherweise sowohl

die nicht-funktionalen Eigenschaften der gesamten Komposition als auch die Komposition selbst. Wobei natürlich auch hier auf die Einhaltung der in der Anfrage vorgegebenen Restriktionen geachtet wird. Sollte auch dieser Versuch den Fehler zu behandeln scheitern, dann kann die Fehlersituation nicht von der ASG-Plattform automatisiert behandelt werden und wird daher als Fehlermeldung an den aufrufenden Service-Consumer weitergeleitet.

Die oben beschriebene dreistufige Fehlerbehandlung dient der Behandlung von technischen Fehlern und nicht in der semantischen Spezifikation eines Services beschriebenen Ausnahmenzuständen. Neben solchen technischen, meistens auf Kommunikationsfehlern basierenden Ausnahmenzuständen können Services auch semantisch-relevante Ausnahmezustände haben. Ein solcher semantisch-relevanter Ausnahmezustand kann zum Beispiel bei dem Service „Abbuchen von Kreditkarte“ der Effekt sein, dass die Kreditkarte ungültig oder abgelaufen ist. Ein anderes Beispiel im Falle des „Paket versenden“ Services wäre der Ausnahmezustand „Paket verloren“. Sollen diese automatisiert behandelbar sein, dann müssen diese als mögliche, wenn auch unerwünschte, Effekte eines Services in der semantischen Service-Spezifikation modelliert werden. In gängigen Prozess- und Workflow-Management Systemen werden idealer Weise alle Ausnahmebedingungen in einem Prozess bzw. Workflow vor ab modelliert. Dieses führt üblicher Weise dazu, dass die Prozesse groß und kompliziert und auf Lange Sicht dadurch schwer wartbar sind. Durch den Ansatz der automatisierten Service-Komposition und der bedarfsgerechten Neuplanung, ist eine andere Lösung für dieses Problem möglich. Die ASG-Plattform geht bei der Planung von Service-Kompositionen von einem optimistischen Szenario aus. Das heißt, dass bei der Planung zunächst lediglich gewünschte Effekte geplant werden. Dadurch werden die Kompositionen kleiner als im Falle einer vollständigen Planung aller möglichen Ausnahmebedingungen, was die Effizienz des Kompositionsvorgangs erhöht. Sollte jedoch während der Ausführung eines Services im Enactment Sub-cycle ein Service nicht einen gewünschten Effekt liefern, sondern in einem semantisch spezifizierten Ausnahmezustand landen, dann wird die teilweise ausgeführte Service-Komposition direkt zum Neuplanen an den Planning Sub-Cycle zurück gegeben. Hier wird dann ausgehend von dem Zustand in dem ein Teil der Komposition erfolgreich bearbeitet wurde und dem Ausnahmezustand des problematischen Services versucht durch Neuplanung der verbleibenden Schritte und Hinzufügen neuer Schritte den Ausnahmezustand zu kompensieren und das alte Ziel weiterhin zu erreichen. Es besteht jedoch auch hier die Möglichkeit des Scheiterns, beispielsweise bedingt durch den Mangel an geeigneten Services die eine Kompensation der Ausnahmesituation ermöglichen oder Aufgrund von Restriktionen des Aufrufs. In diesem Fall bleibt muss auf eine manuelle Lösung für das Problem zurückgegriffen werden, sofern diese überhaupt existiert.

Neben der Ausführung von Services und Service-Kompositionen stellt der Enactment Sub-cycle auch Werkzeuge zum automatisierten Monitoring von Services zur Verfügung. So kann beispielsweise die Zuverlässigkeit von Services im Sinne von Erreichbarkeit des Services und dem erfolgreichen zurückliefern von Resultaten überwacht werden. Diese Monitoring-Daten werden dann über einen Profiler zu einer Bewertung eines konkreten Services oder Serviceanbieters herangezogen. Diese Bewertungsinformationen können auch für die Komposition von Services im Planning Sub-cycle verwendet werden um unzuverlässige Services und Provider in Zukunft zu meiden.

Anwendungsszenarien

Eine Service-Plattform die wie die ASG-Plattform semantisch annotierte Services bereit stellt, dürfte in Zukunft ihre Anwendung in der Integration von Unternehmensinternen und -externen Services sowie als Marktplatz für Services finden. Beim Integrationsszenario wie es in Abbildung 5 skizziert ist, übernimmt die ASG-Plattform die Rolle eines intelligenten Mediators und Integrators, der auf einer hohen Abstraktionsstufe operiert. Services verschiedener interner Anwendungssysteme wie z. B. ERP oder CRM können unter Verwendung der Plattform aufgerufen und zu einem integrierten Anwendungssystem kombiniert werden. Zusätzlich bietet die Plattform die Möglichkeit externe Services transparent in das System zu integrieren und somit für alle internen Systeme verfügbar zu machen. In diesem Zusammenhang sind insbesondere die adaptiven Eigenschaften der ASG-Plattform von Nutzen. Durch die automatisierte Planung und Negotiation von Services können zu jedem Zeitpunkt die jeweils optimalen (z. B. günstigsten) Services genutzt werden. Zusätzlich ermöglicht die mehrstufige Fehlerbehandlungsstrategie der Plattform Ausfälle von einzelnen Services zur Laufzeit zu kompensieren, sofern sich die ausgefallenen Services ersetzen lassen. Die Nutzung von externen Services wird durch die Plattform erleichtert,

weil diese automatisch nach der Registrierung von der Plattform genutzt werden sofern diese in den bestehenden semantischen Kontext der Applikationen passen und zu den anderen, bereits registrierten Services in Bezug auf ihre nicht-funktionalen Eigenschaften „konkurrenzfähig“ sind. Die Registrierung und semantische Adaption der Services und die Bereitstellung einer geeigneten Ontologie liegt bei dem Unternehmen das die Plattform als Integrationswerkzeug verwendet. Dadurch hat das Unternehmen die vollständige Kontrolle über die verwendeten Services was in Bezug auf Sicherheitsaspekte und Vertragsgestaltung von Vorteil ist. Der Betreiber der Plattform kann sich die Serviceanbieter nach Bedarf aussuchen und kann (muss aber nicht) mit Ihnen Verträge in einem gewohnten Rahmen (papierbasiert, nicht notwendigerweise digital) aushandeln und durchsetzen.

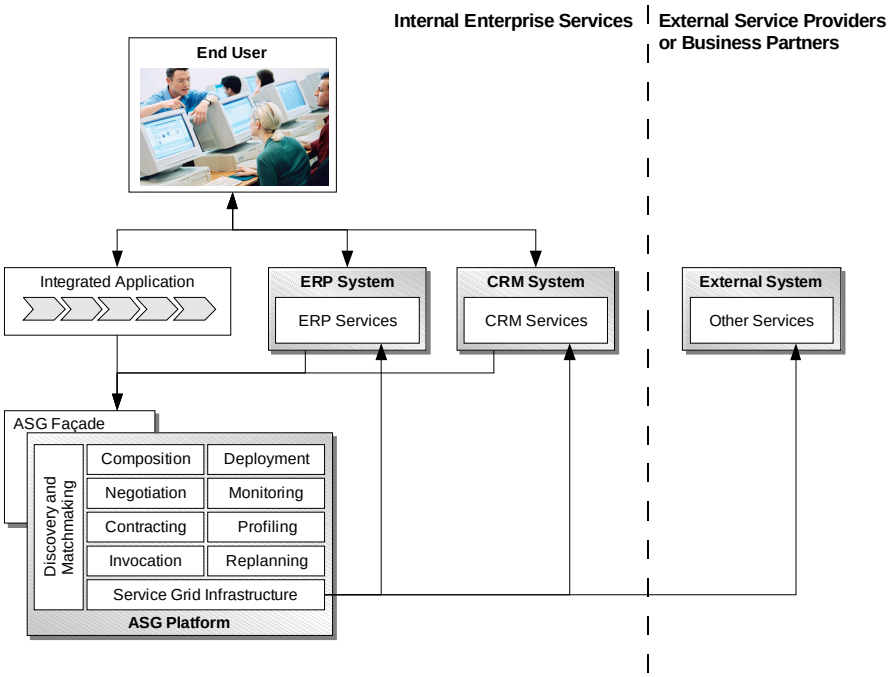

Abbildung 5: Integrationsszenario für ASG-Plattform

Neben dem eher mittelfristigen Einsatzgebiet als Integrationsplattform bietet sich die ASG-Plattform langfristig auch als Marktplatz für Services an (vgl. Abbildung 6). In diesem Szenario gibt es vier beteiligte Gruppen: Die Basic Service Provider stellen grundlegende Dienste zur Verfügung die sie bei dem vom Marketplace Owner betriebenen Marktplatz registrieren. Dieser stellt die Plattform als Marktplatz zur Verfügung und ist für die Wartung derselben und die Bereitstellung einer Ontologie und das Sicherstellen von Vertrauen verantwortlich. Eine wichtige Aufgabe dabei ist eine Überprüfung ob die registrierten Services die von Ihnen angebotene Dienstleistung wirklich anbieten. In diesem Zusammenhang ist insbesondere das von der ASG-Plattform angebotene Monitoring und Profiling von Services von Nutzen. Die ASG-Plattform stellt ihrerseits die in ihr registrierten Services den End Service Providern zur Verfügung. Diese bauen aus den Services der Plattform leichtgewichtige Anwendungen die sie Ihrerseits den institutionellen oder privaten End Service Consumern bereitstellen. Bezüglich einer Implementierung der Vertragsgestaltung zwischen den einzelnen Parteien stehen sich zwei Ansätze gegenüber. Beim ersten Ansatz schließen die Basic Service Provider mit dem Marketplace Owner einen eigenen Vertrag über die Modalitäten der angebotenen Services. Ebenso wie die End Service Provider einen Vertrag mit dem Marketplace Owner über die Nutzung der Services abschließen. In diesem Fall trägt der Marketplace Owner die Haftungsrisiken im Falle einer fehlerhaften Service Ausführung, was jedoch den Vorteil eines höheren Vertrauens in den Marktplatz mit sich bringen dürfte. Alternativ dazu ist es denkbar, dass der Marktplatz lediglich als Mittler für die angebotenen Services dient. In diesem Fall ist es jedoch erforderlich auch eine sichere digitale Vertragsgestaltung zu ermöglichen, damit die End Service Provider zur Laufzeit mit den Basic Service Providern Verträge über die Nutzung von Services abschließen können. Diese Langfristig sicherlich interessantere Alternative stellt zwar hohe Ansprüche an die Sicherheits- und Verschlüsselungs- bzw. Signaturinfrastruktur des Marktplatzbesitzers hat aber den Charme, dass sie eine sehr dynamische und flexible Plattform für elektronische Dienstleistungen ermöglicht.

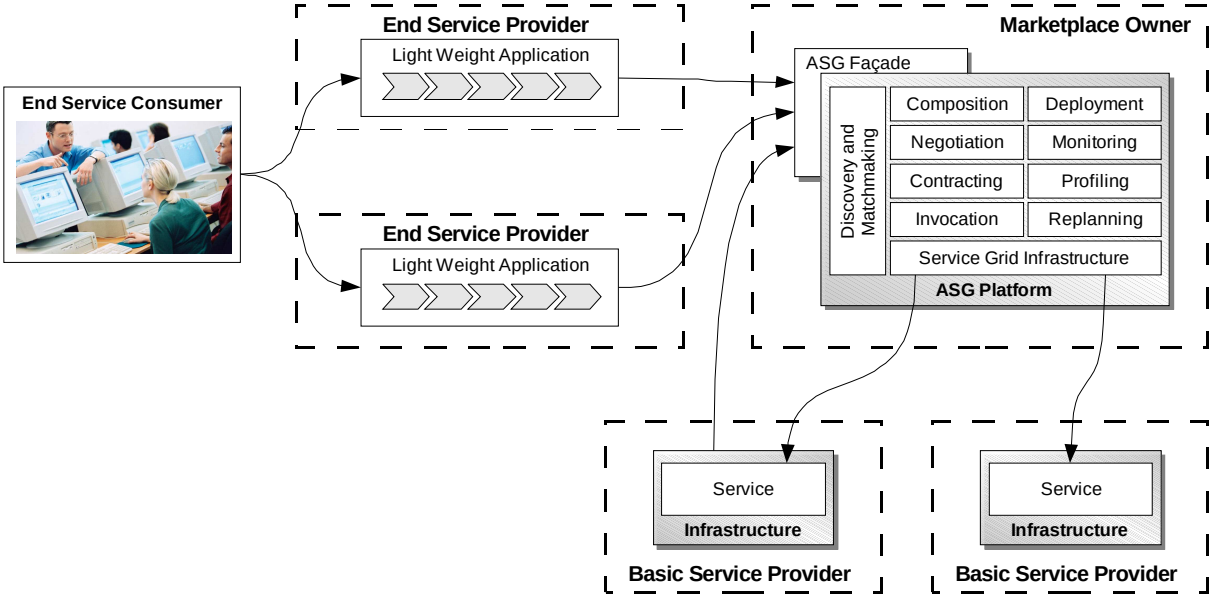

Abbildung 6: Marktplatz-Szenario für ASG-Plattform

Zusammenfassung

In diesem Beitrag wird argumentiert, dass das volle Potential dienstbasierter Softwarearchitekturen nur dann realisiert werden kann, wenn die derzeit verfügbaren syntaktischen Beschreibungen von Diensten um semantische Beschreibungen angereichert werden. Durch diese wird es möglich, konkrete Dienste anhand von Beschreibungen dynamisch zu finden und miteinander zu verknüpfen, um Mehrwertdienste zu realisieren und diese durch Anwendungen dem Markt zur Verfügung zu stellen. Prototypische Plattformen wir das Adaptive Services Grid sind richtungweisende Projekte auf dem Weg das Potential dienstbasierter Softwarearchitekturen vollständig auszuschöpfen. Um dieses langfristige Ziel jedoch für die Praxis anwendbar zu machen, ist es jedoch aus heutiger Sicht eine enge Kooperation zwischen Forschungsinstitutionen und der Softwareindustrie notwendig, um fachspezifische und wissenschaftliche Expertise im Kontext konkreter dienstbasierter Softwarearchitekturen zu kombinieren.

Danksagung: Dieser Artikel enthält Ergebnisse, die im Rahmen des Adaptive Services Grid Projekts (Vertragsnummer: 004617, Identifier: FP6-2003-IST-2) erarbeitet wurden.

Literatur

[1] Andrew Tanenbaum: Modern Operating Systems. 2nd Edition, Prentice Hall, 2001.

[2] Object Management Group: Common Object Request Broker Architecture: Core Specification. Version 3.03. 2004. http://www.omg.org/docs/formal/04-03-01.pdf

[3] World Wide Web Consortium: SOAP Version 1.2 Part 0: Primer. 2003. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[4] Steve Burbeck: The Tao of e-business services. IBM Corporation, 2000. http://www.ibm.com/software/developer/library/ws-tao/index.html

[5] World Wide Web Consortium: Hypertext Transfer Protocol -- HTTP/1.1. 1999. http://w3c.org/Protocols/rfc2616/rfc2616.html

[6] World Wide Web Consortium: Extensible Markup Language (XML) 1.1. 2004. http://w3c.org/TR/2004/REC-xml11-20040204/

[7] World Wide Web Consortium: SOAP Version 1.2 Part 0: Primer. 2003. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[8] World Wide Web Consortium. Web Services Description Language (WSDL) 1.1, 2001.

[9] Organization for the Advancement of Structured Information Standards (OASIS). UDDI Version 2 Specifications, 2002.

[10] Smith, B.: Ontology and Information Systems, 2002. http://wings.buffalo.edu/philosophy/faculty/smith/articles/ontologies.htm

[11] Zúñiga, G.: Ontology: Its Transformation From Philosophy to Information Systems. In Proceedings of Formal Ontology in Information Systems. 2001, pp. 187–197

[12] Kuropka, D.: Modelle zur Repräsentation natürlichsprachlicher Dokumente – Information-Filtering und -Retrieval mit relationalen Datenbanken. In der Serie: Advances in Information Systems and Management Science, 10. Ausgabe. Logos Verlag, Berlin, 2004.

[13] Becker, J.: Integrationsorientierte Wirtschaftsinformatik. http://www.wi.uni-muenster.de/is/aws60.de/becker/modell.htm

[14] de Bruijn, Jos (Editor): The Web Service Modeling Language WSML, 2005. http://www.wsmo.org/TR/d16/d16.1/v0.21/

[15] McGuinness, D; van Harmelen, F. (editors): OWL Web Ontology Language Overview. Web Ontology Working Group at the World Wide Web Consortium (W3C), 2004. http://www.w3.org/TR/owl-features/

[16] World Wide Web Consortium: XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt

[17] Andrews, T; Curbera, F.; Dholakia, H; Goland, Y.; Klein, J; et. al.: Business process execution language for web services version 1.1. Technical report, BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems, 2003.

[18] World Wide Web Consortium: Web Services Activity. http://www.w3.org/2002/ws/

[19] Gesellschaft für Informatik: Informatiklexikon: Web-Services. http://www.gi-ev.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/95/