

Entwurf einer SOA-basierten IT-Service-Architektur für Energiewirtschaften

Jongwoon Hwang

Innovation Research Group
KIST Europe Forschungsgesellschaft mbH
Universität des Saarlandes, Campus E 71
66123 Saarbrücken
hwang@kist-europe.de

Rüdiger Zarnekow

Fachgebiet Informations- und Kommunikationsmanagement
Technische Universität Berlin
Straße des 17. Juni 135
10623 Berlin
ruediger.zarnekow@ww.tu-berlin.de

Abstract: Der mit der Liberalisierung verbundene Wettbewerb auf allen Stufen der Stromversorgung hat deutlich gemacht, dass neue Verfahren zur Kostenoptimierung und Risikosenkung benötigt werden. Dabei stellt die Schaffung einer geeigneten IT-Service-Architektur für die automatische Kommunikation und Kooperation sowie für die Koordination aller beteiligten heterogenen Systeme eine große Herausforderung dar. Der Beitrag beschäftigt sich mit dem Entwurf eines SOA-basierten Architekturmodells für die Umsetzung IT-gestützter Services für das Privatkundengeschäft von Energiewirtschaften. Er beschreibt die hierfür notwendigen Aktivitäten und geht auf die technische Implementierung auf der Basis von Internet-Technologien und Middleware-Ansätzen ein.

1 Ausgangssituation und Motivation

Der verstärkte Wettbewerb, die zunehmende Deregulierung der Energiemärkte sowie die Entwicklung von neuen und innovativen Diensten zwingen Energieversorger dazu, ihre Geschäftsprodukte und -prozesse umzugestalten. Im Mittelpunkt stehen Dienstleistungsangebote, die zu einer Steigerung der Effizienz, einer Verbesserung des Kundenkomforts und einer weitgehend automatisierten Geschäftsführung beitragen. Die damit einhergehende wachsende Bedeutung von Informationstechnologien (IT), die sich von ihrer ehemals unterstützenden Funktion hin zu einem strategischen Erfolgsfaktor wandeln, führt zu steigenden Anforderungen an IT-gestützte Dienstleistungen und stellt viele Energieversorgungsunternehmen (EVU) vor große Herausforderungen.

Insbesondere im Privatkundengeschäft von EVU setzen sich IT-gestützte Dienstleistungen bisher nur zögerlich durch, trotz der vielfältigen technologischen Einsatzmöglichkeiten und der sich daraus ergebenden Wachstumspotenziale. So werden beispiels-

weise die Energieverbrauchsdaten in privaten Haushalten, aber auch großen Unternehmen, meist immer noch mit hohem Aufwand manuell abgelesen und die monatlichen Abbuchungen werden ebenfalls aufgrund von Schätzwerten beziffert [IBM06]. Ursachen hierfür liegen einerseits in der Heterogenität der historisch gewachsenen Informationssysteme mit einer Vielzahl von Schnittstellen und Verknüpfungen und andererseits im begrenzten Anreiz für eine sektor übergreifende Integration der Systeme. Der hierfür erforderliche hohe Investitionsaufwand ist insbesondere für klein- und mittelständische EVU ein kritischer Faktor.

Vor diesem Hintergrund besteht die Hauptzielsetzung dieses Beitrags darin, mögliche neue IT-gestützte Services im Privatkundensegment von EVU aufzuzeigen, und einen Ansatz für deren Konzeption und Umsetzung zu entwickeln. Dies beinhaltet den konzeptionellen Entwurf einer serviceorientierten IT-Architektur¹, die das Verfahren zum Zugriff auf die zu generierenden Servicefunktionalitäten und deren Integration in einer verteilten Systemumgebung beschreibt.

2 IT-gestützte Dienstleistungen im Privatkundengeschäft

Das Geschäft mit Privatkunden stellt eine immer wichtiger werdende Wachstumsquelle für EVU dar. Zugleich besteht in diesem Segment aber auch das größte Risiko, bestehende Kunden an andere Unternehmen zu verlieren. IT-gestützte Dienstleistungen können an dieser Stelle ein zentrales Element einer modernen Kundenbetreuung darstellen. Sie konzentrieren sich auf die zeitnahe Erfassung der Zählerstände, die effiziente Bereitstellung der individuellen Energiedaten sowie eine verbesserte Energieverbrauchsbetreuung. Die Basis bildet ein elektronisches Datenerfassungssystem, das eine exakte Zuordnung der Verbrauchsmenge ermöglicht (siehe Abbildung 1). Ein Datensammlungssystem, welches in regelmäßigen Abständen über eine Netzwerkverbindung mit den Datenerfassungssystemen einzelner Verbraucher oder Verbrauchergruppen kommuniziert, liest die im Datenerfassungssystem gespeicherten Energieverbrauchsdaten ab. Die gesammelten Verbrauchsdaten werden durch ein Datenbanksystem nach Endverbrauchern, verschiedenen Verbrauchergruppen oder nach weiteren vordefinierten Parametern kategorisiert und verschiedenen nachgelagerten Fachanwendungssystemen, wie beispielsweise Abrechnungs-, Controlling-, oder Protokollierungssystem, bereitgestellt. Auf Basis der Verbrauchsdaten kann ein EVU seinen Privatkunden kundenspezifische Informationsdienste vor Ort anbieten. Zu diesen Diensten zählen beispielsweise:

¹ Im Rahmen dieses Beitrags beschreibt die Architektur die notwendigen Zusammenhänge für die Wiederverwendung einzelner Komponenten oder Teilstrukturen während der Gestaltung eines Anwendungssystems. Für eine umfassende Behandlung des Architekturbegriffs wird auf die einschlägige Literatur verwiesen (z.B. [SB01] [FB01] [Rip03] [Bru05]).

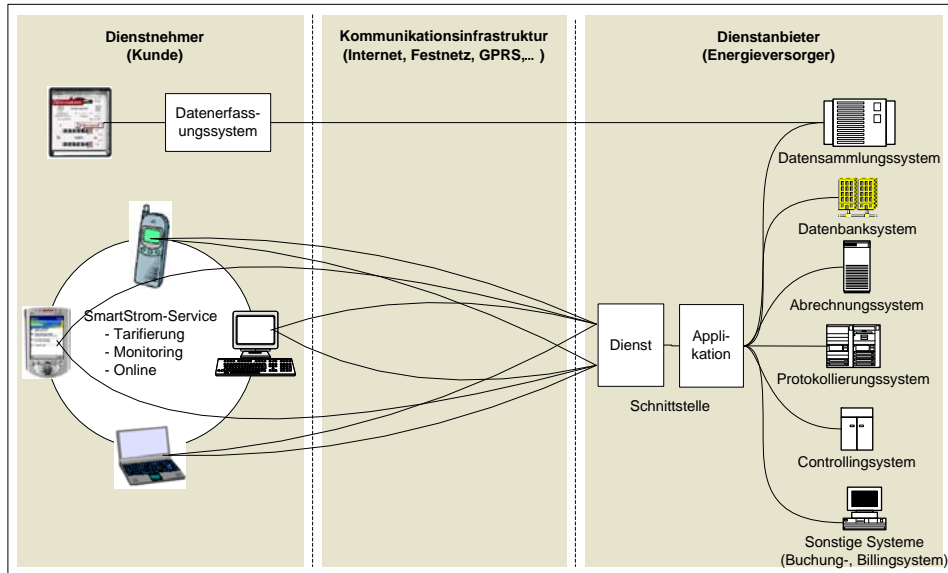


Abbildung 1: Übersichtsdarstellung IT-gestützter Services im Privatkundengeschäft

- **Tarifierungs-Service:** Die gesammelten Verbrauchsdaten werden den jeweiligen Verbrauchern zugeordnet und automatisch und termingerecht elektronische Stromrechnungen erstellt. Die Verbrauchsabrechnung erfolgt mittels bereits existierender Berechnungsmethoden, unter Berücksichtigung verschiedener neu einzuführender Tarifmodelle.
- **Monitoring-Service:** Der Energieverbrauch der Kunden wird dynamisch überwacht, um überproportional ineffiziente Verbraucher zu erkennen. Die Erkennung erfolgt durch einen vordefinierten Maximalwert oder über eine Vergleichsanalyse mit ähnlichen Verbrauchern. Im Falle eines Überschreitens der festgelegten Verbrauchsmenge meldet sich das System automatisch per E-Mail oder SMS beim Kunden.
- **Online-Service:** Die Kunden haben die Möglichkeit, neben allgemeinen Informationen alle persönlichen Energiebezugsdaten wie Rechnungen, aktuelle Zählerstände, historische Energieverbrauchsdaten sowie wichtige Mitteilungen direkt über das Internet einzusehen. Dafür stellt das EVU seinen Kunden eine entsprechende Schnittstelle zur Verfügung. Die Schnittstelle verbindet alle teilnehmenden Systemkomponenten des EVU, um den Kunden eine einheitliche Zugriffsmöglichkeit auf die jeweils verwalteten Dienstobjekte bereitstellen zu können und eine zeit- und ortsunabhängige Interaktion mit den Kunden zu erlauben.

Um umfassende und aktuelle Energiedaten zur Verfügung stellen zu können, müssen verschiedene Informationssysteme eingesetzt und miteinander gekoppelt werden. In der betrieblichen Praxis bestehen dabei Defizite sowohl im mangelnden Zusammenspiel der eingesetzten Systeme, als auch in der Tatsache, dass die den zukünftigen Komponenten zugrunde liegenden Integrationsplattformen noch nicht implementiert sind. Aber auch

wenn die Fachanwendungssysteme miteinander gekoppelt sind, ergeben sich Herausforderungen beim Austausch und bei der Präsentation der heterogenen Daten, da die Unterstützung für standardisierte Datenformate fehlt und die Beschreibungssprachen teilweise stark unterschiedlich ausgeprägt sind.

3 Entwurf einer Service-Architektur

Eine wesentliche Herausforderung für die Realisierung der im vorigen Abschnitt beschriebenen Dienstleistungen besteht darin, die technisch und semantisch heterogenen Bestandteile bzgl. ihrer Funktionen und Datenbestände in eine gemeinsame und einheitliche Plattform zu integrieren. Gleichzeitig ist zu gewährleisten, dass die Systeme miteinander kooperieren können. Hierzu werden im Folgenden die wesentlichen Verfahren zum Datenaustausch zwischen den Komponenten, Anwendungen und Schnittstellen näher betrachtet und deren Implementierung auf der Grundlage einer serviceorientierten Architektur (SOA) beschrieben.

3.1 SOA als Lösungsarchitektur

Es gibt zurzeit verschiedene Ansätze, um Adaptivität in dynamischen Umgebungen zu verbessern und somit verteilte Anwendungen flexibel zu integrieren. Bisherige Internet-Protokolle, die primär auf Interaktionen zwischen Anwendungen und Menschen ausgerichtet waren, weisen bei der Automatisierung der Kommunikationen zwischen maschinellen Komponenten und Applikationen Grenzen auf [MG03]. Beim Enterprise Application Integration (EAI)-Ansatz handelt es sich um eine Lösung, welche sowohl zur Anbindung interner und externer Unternehmensanwendungen als auch zur Modellierung von Geschäftsprozessen genutzt werden kann. EAI stellt aber ein relativ komplexes, unflexibles System dar, das sowohl die Interaktion mit dem Verbindungsadaptor (EAI-Hub) als auch zwischen den Endpunkten der jeweiligen Anwendungen sicherstellen muss [Lor05]. Verschiedene objektorientierte Middleware-Lösungen wie RMI von Sun, CORBA von OMG und DCOM von Microsoft oder Anbindungstechnologien wie CGI oder PHP erreichen zwar im übergeordneten Ablauf eine gewisse Flexibilität, jedoch ohne die Probleme der eingriffslosen Wiederverwendbarkeit, der semantischen Spezifikation und der aufwendigen Anpassung der nichtfunktionalen Anforderungen befriedigend zu lösen [Hof05]. Darüber hinaus haben sie zumeist nicht zum Ziel, Dienste extern anzubieten, sondern beschäftigen sich primär mit der organisationsinternen Kopplung von Anwendungen. Nachteil ist die oftmals enge Kopplung von Komponenten, die auf eine mangelnde Standardisierung der Technologien zurückzuführen ist [DJ04].

Eine SOA löst diese Probleme durch die Kombination von Internet-Technologien und Middleware-Ansätzen auf Basis von XML als Datenformat und SOAP als Kommunikationsprotokoll [Sta02]. Beide Technologien sind plattform- und programmiersprachenunabhängig und garantieren damit höchste Interoperabilität. Im Gegensatz zu den eng gekoppelten Kommunikationsarchitekturen herkömmlicher Middleware, bei denen ein großer Abstimmungsaufwand durch eine Neuausrichtung der Kommunikations-

umgebung entsteht [Coy02], konzentriert sich die SOA auf die Beschreibung und Verwaltung von Diensten, sodass sie zur Laufzeit gesucht, dynamisch gebunden und schließlich genutzt werden [Gie02]. Durch die Erweiterung der Infrastruktur auf Sektor übergreifende Netzwerke werden die Vorteile dieses Ansatzes mit einer effizienten Allokation und Technologie übergreifenden, herstellerunabhängigen Kooperation von Service-Ressourcen verbunden. Die einzelnen Services der Komponentenbauweise können kurzfristig, dynamisch und simultan zu unterschiedlichen neuen 'Super-Services' zusammengefasst und an die jeweiligen Bedürfnisse angepasst werden [Wij01]. Weiter umfasst das SOA-Konzept ein organisatorisches Rahmenwerk, das ausschließlich auf einer Reihe von etablierten Standards basiert. Damit ist es vollkommen unabhängig von Softwareherstellern, Middleware-Plattformen und Programmiersprachen.

3.2 Standards zur Umsetzung von SOA

Die im Folgenden beschriebene SOA für den Aufbau und Betrieb IT-gestützter Dienstleistungen in EVU setzt auf einer Reihe von webservicebasierten Middleware-Technologien auf. Mit dem Webservice-Konzept besteht eine realistische Möglichkeit, auf der Middleware-Ebene unter Nutzung standardisierter Protokolle und Sprachen zu einer allgemein akzeptierten Lösung für die Integration von Internet-basierten Softwarekomponenten zu gelangen [MFR03]. Dabei sind die Dienste unabhängig von der jeweils eingesetzten Technologie. Implementierungsdetails werden gekapselt und bleiben dem Dienstanrufer verborgen. In der Regel wird eine Webservice-Anwendung anhand von WSDL-Dokumenten beschrieben. Dazu bietet der Dienstanbieter einen standardisierten Zugriff auf seine Anwendung an und macht eine Beschreibung der von ihm angebotenen Dienste öffentlich über ein Dienstregister verfügbar. UDDI stellt hierbei das Dienstregister dar, das die durch die Webservices zur Verfügung gestellten Dienste verwaltet. Der Dienstnehmer kann einen Dienst in Anspruch nehmen, indem er zunächst den entsprechenden Dienst über das Dienstregister ausfindig macht und dann mit dem Dienstanbieter entsprechend den in der Dienstbeschreibung angegebenen Protokollen und Schnittstellen kommuniziert. SOAP ist ein XML-basiertes Kommunikationsprotokoll für Webservice-Anwendungen, das den Nachrichtenaustausch zwischen unterschiedlichen Plattformen realisiert. Die benötigten Technologien und Spezifikationen, welche zusammen die Anforderungen einer SOA abdecken, können mithilfe eines Schichtenmodells oder sog. Webservice-Stacks, dargestellt werden (vgl. Abbildung 2).²

² Es gibt keine einzelne, konkrete Webservice-Technologie, vielmehr werden Webservices stets durch ein Bündel komplementärer Technologien realisiert [KBSt05]. Der hier dargestellte Stack ist aber nur eine von vielen Varianten eines Webservice-Stack. Die Vielzahl unterschiedlicher Varianten lässt sich durch die verschiedenen Interessen und Betrachtungsweisen, unter denen ein Stack betrachtet werden kann, erklären [DJM05].

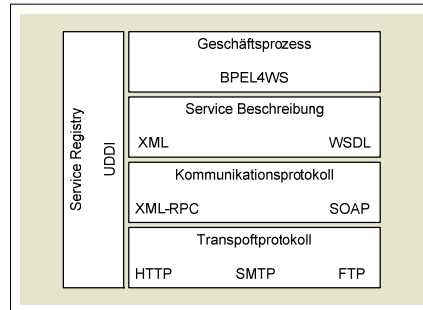


Abbildung 2: Webservice-Protokoll Stack

3.3 Aufbau der Service-Architektur anhand des Schichtenmodells

Im Rahmen des Architekturentwurfs sind die folgenden Aktivitäten vorzunehmen:

- **Einordnung und Zugriff der Systemkomponenten**

Ein zentrale konzeptionelle Aufgabe zum Entwurf der Service-Architektur besteht in der Kombination und Integration unterschiedlicher Dienste, die von verschiedenen Fachanwendungssystemen angeboten werden. Die einzelnen Dienste können selbst auch komplexe Strukturen enthalten. Diese Komplexität wird aber durch die klare Definition von Dienstoperationen und -schnittstellen gekapselt und bleibt somit verborgen. Sie überwindet Unterschiede auf den einzelnen Ebenen der Interaktion und bestimmt zusammen mit den entsprechenden Komponenten in einer konkreten Systemtopologie die gesamte Integrationslösung. Dieser modulare Aufbau schafft Flexibilität und Erweiterbarkeit, einzelne Dienste können ersetzt oder neu komponiert werden, ohne dass die Gesamtarchitektur bzw. der Gesamtprozess geändert werden muss. Durch die Bildung wiederverwendbarer Dienste aus Anwendungen und Datenobjekten können die beteiligten Komponenten modular zusammengestellt und daher auch einfach gesteuert werden. Somit können die bereitgestellten Inhalte, Dienste und Funktionen beliebig konfiguriert und zudem nutzerspezifisch angepasst werden. Das Zusammenwirken der einzelnen Komponenten und die Dienstklassen werden in Bezug auf Java-EE-Spezifikationen in Abbildung 3 anschaulich dargestellt. Grundlage hierfür bildet eine Multi-Tier-Architektur.

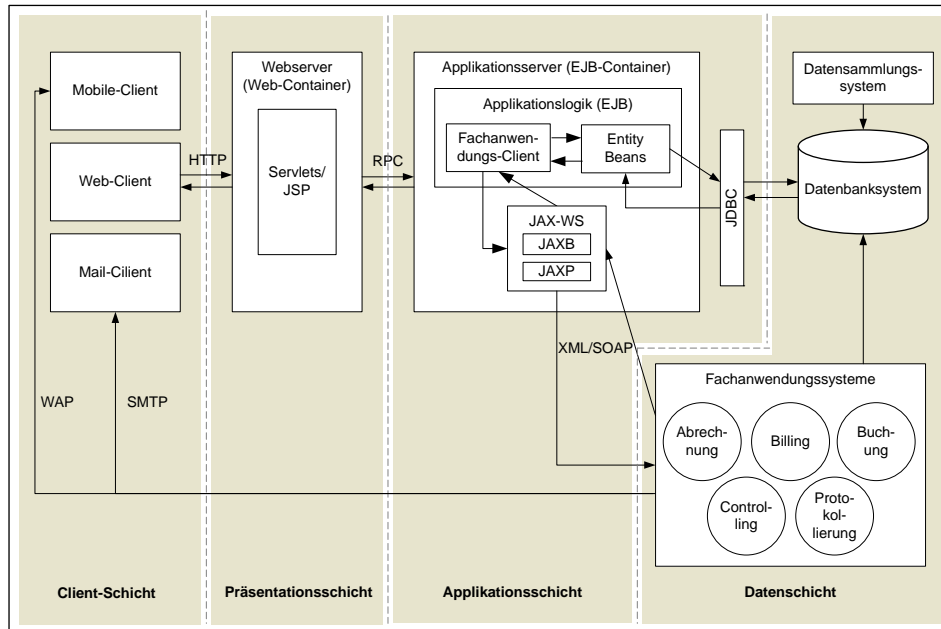


Abbildung 3: Service-Architektur auf Basis einer Java-EE-Plattform

Der Webserver bildet den Backbone für die Schnittstelle, die zur Annahme, Verarbeitung und Beantwortung von HTTP-Anfragen dient. Die Applikationslogik der Fachanwendungen wird mit Hilfe von EJBs implementiert. Sie beinhalten die zur Realisierung des Dienstes nötigen Methoden, welche Clients des EJB-Containers über einen RPC-Mechanismus ausführen können. Die Anbindung der Fachanwendungen an die Dienstplattform wird über Webservices realisiert. Die Komponenten, die zur Nutzung von Webservices benötigt werden, sind die WSDL zur Beschreibung der Schnittstellen und das Protokoll SOAP für den Datenaustausch. Beides wird von den in der Spezifikation JAX-WS spezifizierten Komponenten bereitgestellt. Da Webservices auf einem Datenaustausch im XML-Format basieren, benötigt JAX-WS weitere Komponenten, um Datenobjekte in XML zu verwandeln sowie in der anderen Richtung aus XML-Fragmenten Datenobjekte zu generieren. Diese Funktionalität ist in JAXB beschrieben, die benötigten XML-Parser sind in JAXP spezifiziert. Die Datenbank stellt eine grundlegende Datenbasis für die fachspezifischen und fachneutralen Anwendungen zur Verfügung. Die Datenbankanbindung der Dienstplattform wird mit Hilfe sog. Entity-Beans realisiert, die im EJB-Container laufen. Außerdem umfasst der Datendienst der Plattformarchitektur die Datensynchronisation im Bezug auf die dienstbezogene Verteilung der fachspezifischen Anwendungen.

- **Integration der Webservice-Schnittstelle**

Voraussetzung für die Integration autonomer Fachkomponenten in eine einheitliche IT-Umgebung ist, dass die Fachanwendungssysteme bereits über eine standardisierte

Webservice-Schnittstelle verfügen, welche eine XML-basierte Dienstbeschreibung und Datentransformation unterstützt. Die Webservice-Schnittstelle wird mithilfe eines WSDL-Dokuments spezifiziert, damit ein Webservice-Client die Anwendung nutzen kann. Die WSDL-Datei spezifiziert dabei nicht nur die Methoden, die aufgerufen werden können, sondern sie macht auch Angaben über die konkrete Realisierung der Services. Da die Methoden mit Eingabe- und Ausgabeparametern ausgestattet sind, müssen auch diese definiert werden.

Im Folgenden wird der Aufbau eines WSDL-Dokuments, das einen möglichen Tarifierungs-Service beschreibt, beispielhaft anhand der Definition der Methode *berechneRechnung* aufgezeigt. Im message-Element werden die Nachrichten festgelegt, die durch die Webservices beim Aufruf der Operation ausgetauscht werden. Zu dieser Beschreibung gehören die Datentypen der dabei ggf. transportierten Methodenparameter und Rückgabewerte. Diese Nachrichten kapseln die Parameter der Webservice-Methode, getrennt nach Eingabe- und Rückgabeparametern. Die verwendeten Datentypen stammen entweder aus der XML-Schema-Definition oder wurden im types-Element deklariert (vgl. Listing 1).

```
<message name = "berechneRechnungRequest">
  <part name = "RechnungID" element = "xsd:string" />
</message>

<message name="berechneRechnungResponse">
  <part name = "Betrag" element="xsd:double" />
</message>
```

Listing 1: Definition von Datentypen in der WSDL-Beschreibung

Im portType-Element werden die Definitionen der message-Elemente verwendet, um die Signatur der Webservice-Methoden zu beschreiben (Funktionsname, Eingabeparameter, Rückgabewerte). Im Beispiel wird die Methode *berechneRechnung* mit den Elementen der Nachricht *berechneRechnungRequest* als Eingabeparameter und den Elementen der Nachricht *berechneRechnungResponse* als Rückgabewerte definiert (vgl. Listing 2).

```
<portType name="BillingPortType">
  <operation name="berechneRechnung">
    <input message="tns: berechneRechnungRequest" />
    <output message="tns: berechneRechnungResponse " />
  </operation>
</portType>
```

Listing 2: Definition eines Port-Typs aus Operationen in der WSDL-Beschreibung

Im Anschluss kann die Schnittstelle über konkrete Protokolle an ihre Implementierung gebunden werden. Das binding-Element beschreibt die Bindung der portType-Elemente an Protokolle (z. B. SOAP, HTTP GET/POST und MIME) durch die Definition des Nachrichtenformats und legt somit auch die Protokolle fest, die der Webservice interpretieren kann. Im Beispiel ist der Tarifierungs-Service in der Lage, die Kommunikation zur Ausführung der als *berechneRechnung* definierten Methode über das SOAP-Protokoll abzuwickeln. Listing 3 setzt das Beispiel fort.

```

<binding name="BillingSOAPBinding" type="tns:BillingPortType">
<soap:binding style="rpc"
  transport="http://schemas.xmlsoap.org/soap/http" style="document" />

  <operation name="berechneRechnung">
    <soap:operation soapAction="" />
    <input>
      <soap:body use="encoded" namespace="Billing"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>

    <output>
      <soap:body use="encoded" namespace="Billing"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>

```

Listing 3: Definition der Binding in der WSDL-Beschreibung

- **Ausführung des Datenmapping**

Damit ein in WSDL beschriebener Dienst in Java implementiert oder aufgerufen werden kann, müssen entsprechende Java-Klassen und Datenstrukturen vorhanden sein, die den in WSDL beschriebenen Informationen entsprechen. Außerdem müssen sowohl die WSDL-Elemente auf Java (Stub) als auch die Java-Konstrukte auf WSDL-Elemente (Skeleton) abgebildet werden. Abbildung 4 zeigt die Verwendung von WSDL am Beispiel eines Aufrufs der vom Tarifierungs-Service bereitgestellten Operation in Java Syntax: *public int berechneRechnung(String Rechnung_ID)*. Der Codeausschnitt in der Abbildung stellt den generierten Code für einen clientseitigen Aufruf dieser Methode dar. Der Webservice-Client verwendet dabei generierte Klassen, um die Webservice-Methoden zu nutzen. Zur Vereinfachung der Implementierung dieser Ausführung können mit einem Tool wie Axis, das an der JAX-WS Spezifikation ausgerichtet ist, automatisch Java-Klassen von WSDL-Dateien erzeugt werden. Axis wurde entwickelt, um auf Web- oder Applikationsservern installiert zu werden und kann von den meisten verbreiteten Servern (z. B. Tomcat, JBoss, WebSphere) ausgeführt werden. Sowohl bei der Generierung von clientseitigen Stubs und serverseitigen Skeletons aus einem WSDL-Dokument als auch bei der Rekonstruktion von Methodenaufrufen und Rückgabewerten aus einer SOAP-Nachricht müssen XML-Daten in Java-Objekte umgewandelt werden.

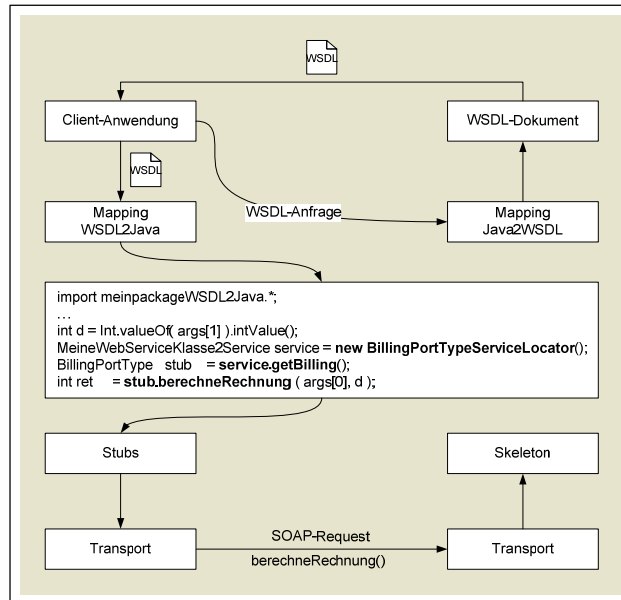


Abbildung 4: WSDL2Java-Mapping in der Webservice-Anwendung

3.4 Erweiterung der Service-Architektur

Alternativ zu der im Service-Szenario dargestellten Architektur, bei der alle seitens des Energieversorgers beteiligten Teilsysteme auch vom EVU selbst betrieben werden, ist es vorstellbar, dass die Geschäftsprozesse entweder innerhalb der unternehmensinternen Infrastruktur verwaltet werden oder auf die verschiedenen Standorte über Systemgrenzen hinweg ausgelagert werden können. Ein EVU kann beispielsweise die Fachanwendungen Tarifierungs-Service und Monitoring-Service jeweils an externe IT-Dienstleister auslagern, die dann die entsprechenden Dienste per Webservices anbieten. Weiterhin wird ermöglicht, dass die Datenbank für die kundenbezogenen Stammdaten (Personaldaten, Tarifmodell, Verbrauchsdaten usw.) aus Performance- und Sicherheitsgründen im EVU-internen Netzwerk angesiedelt ist, während die fachanwendungsspezifischen Daten (Beträge, Rechnung, Konto usw.) innerhalb der Infrastruktur der IT-Dienstleister gespeichert sind. Zum Zugriff auf die Stammdaten erhalten die Dienstleister jeweils eine eigene, automatisch replizierte Kopie der Stammdatenbank. Abbildung 5 zeigt schematisch einen Überblick über die Dienstarchitektur sowie das Zusammenspiel der einzelnen Komponenten. Im Rahmen der SOA-Konzepte ist es unabhängig von der geografischen Verteilung der Systeme wesentlich, zu welchen Organisationseinheiten des Diensteanbieters im Zuge der Realisierung einer Interaktionsschnittstelle Kommunikationsbeziehungen etabliert werden müssen, welche Datenstrukturen dabei auszutauschen sind und welche Schnittstellen von den beteiligten Organisationseinheiten jeweils ausgeprägt werden müssen.

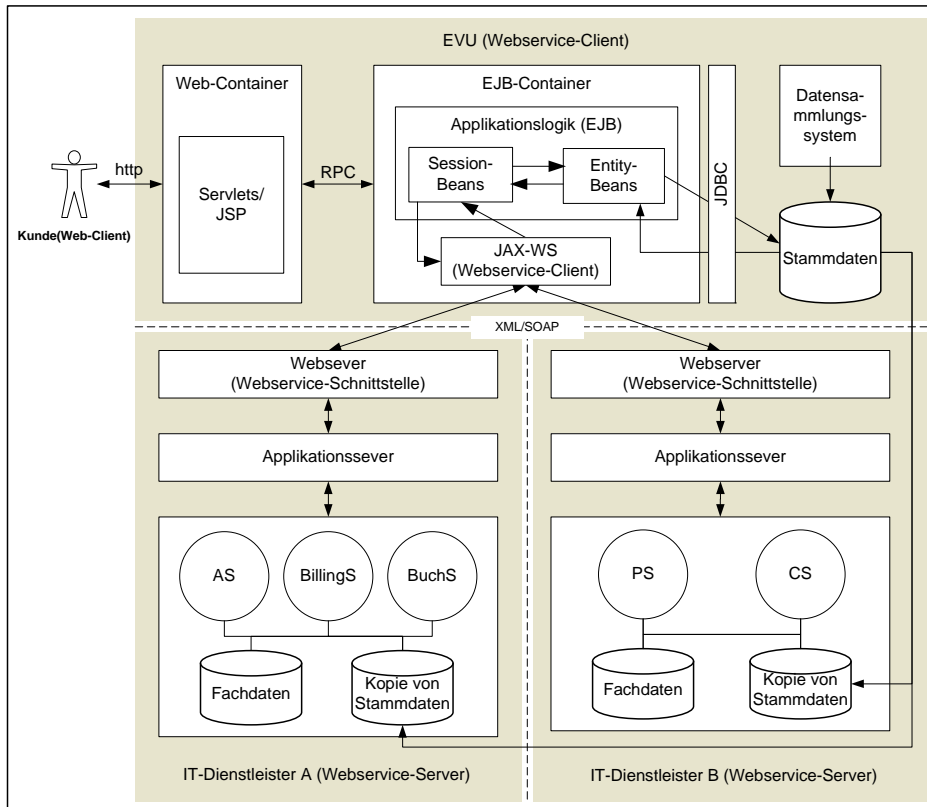


Abbildung 5: Erweiterung der Service-Architektur in Bezug auf Java-EE-Plattform

4 Zusammenfassung und Ausblick

Der Beitrag zeigt einen Lösungsansatz auf, mit dem die Heterogenität von Informations- bzw. Ereignisquellen im Privatkundensegment von EVU adressiert und eine Vielzahl von verschiedenartigen Systemkomponenten über ein Kommunikationsnetzwerk in einer einheitlichen Arbeitsumgebung zusammengefasst werden können. Dazu wurde zunächst betrachtet, mit welchen Herausforderungen sich die heutigen Versorgungswirtschaften konfrontiert sehen. Auf Basis dieser Analyse wurde ein mögliches Service-Szenario im Rahmen der privatkundenbezogenen Energieverbrauchsbetreuung generiert. Die Lösungsidee lag in der Herstellung einer webservice-basierten Schicht oberhalb der bisherigen Middleware-Ansätze in Form von mehrfachen Verknüpfungen der verteilten Teilleistungen. Damit wurde eine Infrastruktur zur Strukturierung der einzelnen Module und ihrer Beziehungen bereitgestellt. Die Umsetzung IT-gestützter Dienstleistungen für EVU wird in der Zukunft an Bedeutung gewinnen. Die anhaltende Verbreitung der Datennetze sowie der kontinuierliche Rückgang der Transaktionskosten führen in der Versorgungsbranche dazu, eine praktische Relevanz nicht nur für die Prozess-

optimierung und Erhöhung der Flexibilität, sondern auch für die Stärkung der Kundenbindung und Erschließung neuer Märkte zu erkennen. Hierdurch sind die ökonomischen und technischen Voraussetzungen dafür gegeben, nicht nur große Stromkunden, sondern auch kleinere Industrie- und Haushaltskunden in eine IT-gestützte Erbringung von Dienstleistungen mit einzubeziehen [FCK06].

Literaturverzeichnis

- [Bru05] Brugger, R.: IT-Projekt strukturiert realisieren, 2. Auflage, Vieweg Verlag, Wiesbaden, 2005
- [Coy02] Coyle, F.P.: XML, Web Services, and the Data Revolution, Addison-Wesley, 2002
- [DJ04] Dostal, W.; Jeckle, M.: Semantik, Odem einer Service-orientierten Architektur. in: Javasppektrum, 2004, S. 53-56
- [DJM05] Dostal, W.; Jeckle, M.; Melzer, I.; Zengler, B.: Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis, Spektrum Verlag, München, 2005
- [FB01] Foegen, M.; Battenfeld, J.: Die Rolle der Architektur in der Anwendungsentwicklung, in: Informatik-Spektrum 24, 2001, S. 290-301
- [FCK06] Franz, O.; Cremer, C.; Kimpeler, S.; Schäffler, H.: Potenziale der Informations- und Kommunikationstechnologien zur Optimierung der Energieversorgung und des Energieverbrauchs (eEnergy), Studie für das BMWi, wik-Consult – FhG Verbund Energie, Bad Honnef, 2006
- [Gie02] Gieß, M.: Architektur für transaktionssichere Web-Service-Anwendung, Diplomarbeit, Technische Universität München, Institut für Informatik, 2002.
- [Hof05] Hofmann, O.: Analyse und Test verschiedener Ansätze zur Beschreibung und Komposition von komplexen Web Service, Bachelorarbeit, Technische Universität Dresden, Fakultät Informatik, 2005.
- [IBM06] o. V.: Branche unter Hochspannung, in: Energiemorgen, IBM Deutschland GmbH, München, 2006.
- [KBSt05] o. V.: Interoperabilität von Office-Anwendungen auf Basis von XML, KBSt der Bundesregierung für Informationstechnik in der Bundesverwaltung, 2005
- [Lor05] Lorenziell-Scholz, D., Service-orientierte Integration mit einem 'Enterprise Service Bus', 2005, S. 19-23
- [MFR03] Mehl, O.; Feuerhelm, D.; Rudin, T.; Abeck, S.: Ein Web-Service-basierter Dienst für den homogenen Zugriff auf Wissensmaterialien, in: Kommunikation in verteilten Systemen (KiVS), GI/ITG-Fachtagung, Leipzig, 2003.
- [MG03] Malek, M.; Günther, O.: Architekturen und Geschäftsmodelle, Beschreibung des Teilprojektes, Humboldt Universität zu Berlin, Ein Teilprojekt von: InterVal – Internet Value Chains, 2003.
- [Rip03] Riempp, G., Integrierte Wissensmanagement-Systeme, Springer Verlag, Berlin, 2003
- [SB01] Stafford, J. A.; Wolf A. L.: Software-Architecture. in: Component-Based Software Engineering - Putting the Pieces Together, Heineman G. T.; Councill W. T. (Hrsg.), Addison-Wesley, Boston, 2001, S. 371-387
- [Sta02] Stal, M.: Web Services: Beyond Component-Based Computing, CACM, Vol 45, No.10, 2002, S. 71-76
- [Wij01] Wijnhoven, F.: Models of Information Markets - Analysis of Markets, Identification of Services and Design Models, in: Informing Science - Special Series on Information Exchange in Electronic Markets, Vol.4, 2001, S. 117-128